SONDERHEFT 75 Markt&Technik öS 120,-/sfr. 16,-/Lit. 16000 hfl. 21,-/dkr. 75,-/fmk 62,- DM 16,-

DISKETTE IM HEFT

64'er

Grafik

Über 40 Programme auf Diskette

Mal- und Zeichenprogramme

Hi-Eddi jetzt mit 3-D-Grafik

Doppelte Auflösung

640 x 200 - Pixel durch Interlace 64

Raffinierte Profitricks

So programmieren Sie Demo-Intros

Beeindruckende Fraktale

Selbermachen: Bizarre Computerlandschaften

INHALT



Erweiterungen

Rosarote Brille
»Interlace 64«: 136 Farben, doppelte Bildschirmauflösung von z.B. 640 x 200 Pixeln, 80-Zeichen-Textmodus - mehr kann man vom C64 kaum noch verlangen! **4**

Geometrisches Heimkino
»Blitter«: Das ausgefuchste Animationsprogramm entwirft zweidimensionale Körper und läßt sie über den Bildschirm huschen - wie im Kino **7**

Schildkröte als Sprinter
»Turtle Graphics«: das ideale Tool, um Hires-Grafiken schnell mal zwischendurch zu entwerfen. **10**

Bizarre Computerlandschaft
»Superfrac 64«: Was hat der Grand Canyon mit Mathematik zu tun? Darüber klärt Sie unser Fraktal-Programm auf. **14**

Zeichensatz

Verschnörkelt oder cool
»Charmaster«: Super-Zeichensätze zum Selbermachen, ohne Frust und Quälerei. **15**

Sprites

Spritzige Sprites
»Spritemon 2.2«: gehört zur Grundausstattung jedes Programmierers, um die tollsten Effekte auf den Bildschirm zu zaubern. **24**

Moving Pictures
»Sprite-Moover«: haucht den imaginären Monstern und Fabelwesen Leben ein. **26**

Grundlagen

In neuem Gewand
Deutsche Umlaute und Sonderzeichen, Hieroglyphen oder mehrfarbige Spiele-Levels: Mit einem geänderten Zeichensatz ist alles möglich! **28**

Premiere für hohe Auflösung
Hires-Grafik läßt sich nur per Maschinensprache effektiv zur Schau stellen. Wir liefern Ihnen die Tools und Utilities, die man in eigenen Basic-Programmen verwenden kann **32**

Grafik-News

Effekthascherei
Wie eine Suppe ohne Salz - ein Computerspiel ohne fetziges Intro. Profis zeigen, wie man's richtig macht. **36**

Zeichenprogramm

Louvre im C64
»Hi-Eddi«: Würde Leonardo da Vinci heute leben, wäre er von diesem Mal- und Zeichenprogramm begeistert. Sogar 3-D-Animationen sind für dieses Tool kein Problem. **40**

Tips & Tools

Praktische Tiefkühlkost
»Sir-Freeze«: friert alle Hires-Grafiken und Sprites auf dem Screen ein und speichert sie auf Diskette. Grafikklau in Perfektion! **46**

Tiefstapler mit Pfiff
Sieben kurze, aber umso effektvollere Utilities bereichern Ihre Tool-Bibliothek. Egal, ob Sie Sprites suchen oder z.B. Text nachträglich in einer Hires-Grafik unterbringen möchten. **47**

Sonstiges

Diskettenseiten **18**

Impressum **20**

Leserumfrage
Machen Sie mit - wir verlosen zehn 64'er-Sonderhefte unter den Einsendern! **22**

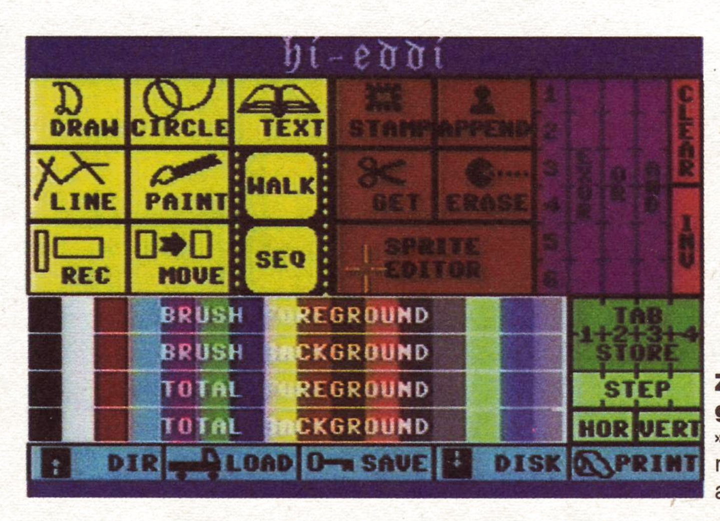
Disklader **23**

Vorschau **50**

Alle Programme zu Artikeln mit einem ■-Symbol finden Sie auf der beiliegenden Diskette (Seite 19).



Der Trick mit der Rasterzeile: Tanzende Schriftzeichen und sekundenschneller Farbwechsel zeichnen professionelle Intros aus **Seite 36**



Zeichenprogrammklassiker: »Hi-Eddi« kann noch mehr, als viele ahnen! **Seite 40**



Interlace 64 - hochauflösende Grafik

Wollten Sie schon immer mehr auf den Bildschirm bringen als der C64 bietet? Beispielsweise 136 Farbkombinationen oder doppelte Bildschirmauflösung. Dann ist »Interlace 64« genau das richtige.

Eine Schwäche des C64 ist die Darstellung von nur 40 Zeichen in einer Zeile, bringt doch jeder Drucker mindestens 80 Zeichen zu Papier. An so manchem Trick wurde getüftelt. Überzeugen konnte allerdings keine der (Software-)Lösungen, denn leider lassen sich, egal in welchem Grafikmodus, nur max. 320 Punkte horizontal auf einem Bildschirm darstellen. Will man also 80 Zeichen erreichen, reduziert sich die Breite auf vier Punkte. Damit sind leider nicht alle Buchstaben und Sonderzeichen in Ihrer richtigen Form darstellbar.

Wir stellen Ihnen eine neue Methode vor, natürlich zusammen mit dem entsprechenden Programm, mit dem diese Ärgernisse abgestellt sind: »Interlacing« (Abb.1 und Abb 3). Dieses Verfahren verwendet zwei Grafikbildschirme, auf denen jeweils eine Hälfte des Bildinhalts sichtbar ist. Dafür wird der erste Schirm mit den geraden Bits gefüllt (0, 2, 4, 6) der zweite mit den ungeraden (1, 3, 5, 7). Schaltet man nach einem kompletten Bildschirmaufbau (also jede 25stel Sekunde) den Bildschirm wechselweise um, erreicht man die doppelte Pixelanzahl (640 x 200 Punkte oder 80 Zeichen). Als Preis dafür tritt allerdings ein geringfügiges Flackern auf. Da zugleich die Hintergrundfarben gewechselt werden, sind jeweils zwei Farbkombinationen möglich (16 x 16 Farben = 256 Kombinationen = 136 unterschiedliche Farben. Natürlich funktioniert dieses Prinzip auch in vertikaler Richtung (Abb. 2). Hier tauscht man lediglich die geraden Linien (0, 2, 4, 6 usw.) mit den ungeraden (1, 3, 5, 7 usw.) und erreicht damit doppelte vertikale Auflösung (320 x 400 Punkte oder 50 Zeilen). Ein Demo-Programm für diese Modi finden Sie auf der beiliegenden Diskette. Sie laden es mit

```
LOAD"SPLIT.DEMO",8
```

und starten mit RUN. »Split.Demo« zeigt alle Variationen des Interlacing auf einem Bildschirm. Wenn Sie sich ausreichend informiert haben, sollten Sie den C64 einmal aus- und einschalten. Die Basic-Erweiterung laden Sie mit

```
LOAD"INTERLACE 64",8
```

Danach starten Sie mit RUN. Beachten Sie, daß <RUN/STOP RESTORE> zur Unterbrechung des Programms führt, seine Funktionen lassen sich danach nicht mehr nützen. Ein Neustart ist aber mit SYS49152 möglich.

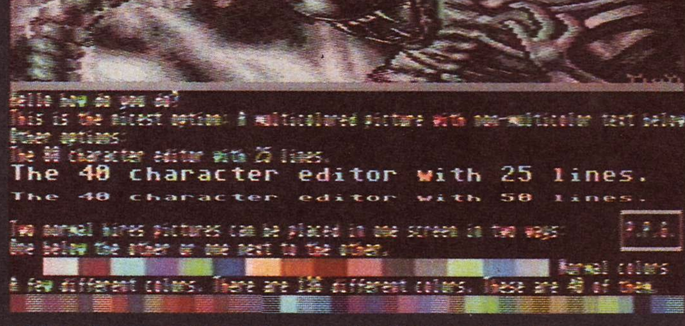
Nach dem Programmstart stehen Ihnen drei Bildschirmauflösungen zur Verfügung. Wie bei allen anderen Befehlen gibt es mehrere Wege sie anzuwählen. Fast alle Kombinationen lassen sich für eigene Programme innerhalb von PRINT-Anweisungen in Anführungszeichen übernehmen oder mit »CHR$(X)« erreichen. Lediglich bei den Tastenkombinationen <CTRL CBM und Taste> funktioniert dies nicht. Verwenden Sie in diesem Fall die »←-Codes«.

Bildschirmauflösungen

<CTRL D> - CHR$(4)
... schaltet um auf 50 Zeilen/ 40-Zeichen-Darstellung.

<CTRL U> - CHR$(21)
... wechselt auf 25 Zeilen/ 80-Zeichen-Darstellung.

<CTRL Y> - CHR$(25)
... führt zur 25 Zeilen/ 40-Zeichen-Darstellung (normaler Modus).



[1] Das Demo zeigt einige der 136 Farbkombinationen und Schriftmodi von »Interlace 64«

Dieser Code läßt sich auch in einem Basic-Programm (innerhalb Anführungszeichen) in einer PRINT-Anweisung verwenden.

Beachten Sie, daß für eine akzeptable Darstellung der beiden Interlace-Modi ein Monitor angeschlossen sein muß. Ein Fernseher ist untauglich.

Zunächst werden Sie kaum eine Veränderung zum normalen Betriebssystem feststellen, lediglich die PRINT-Ausgaben sind langsamer als sonst. Wenn Sie mit <CTRL D> auf 50 Zeilen/ 40 Zeichen umschalten, halbiert sich die Höhe des Cursors und die Ausgaben sind am Bildschirm nur noch halb so hoch - bei <CTRL U> entsprechend nur halb so breit. <CTRL Y> schaltet zurück auf die normale Darstellungsform des C64. Beachten Sie, daß der Cursor bei jedem Moduswechsel zurück zur ersten Bildschirmzeile springt und der Textspeicher gelöscht wird. Damit sehen Sie zwar die vorherige Darstellung noch auf dem Bildschirm, aber Eingaben lassen sich nicht mehr mit <RETURN> bestätigen. Beispiel:

Sie haben »LIST« im Modus 25 Zeilen/ 40 Zeichen eingegeben und schalten auf 25 Zeilen/ 80 Zeichen um. Dann steht zwar noch LIST am Bildschirm, aber wenn Sie mit dem Cursor über diesen Befehl fahren und mit <RETURN> bestätigen, passiert nichts, da der Textspeicher gelöscht wurde.

Das Positionieren eines Textes wird durch einen speziellen Befehl erleichtert:

```
←z,x,y,"TEXT"
```

Wobei »←z« den Befehl einleitet, »x« ist die horizontale Position. Je nach Modus sind dabei Werte von 0 bis 39 oder 0 bis 79 erlaubt. »y« gibt die Zeile an, in der die Textausgabe erfolgen soll. Hier dürfen Sie entsprechend dem Modus die Werte 0 bis 24 oder 0 bis 49 verwenden. »"TEXT"« steht für das, was ausgegeben werden soll. Für X, Y und "TEXT" las-

sen sich Variable verwenden. Beispielsweise setzt
`A=10:B=11:A$="TRALLALA":←z,A,B,A$`
den Text »TRALLALA« auf Spalte 10 in Zeile 11.

Textausgabe für verschiedene Bereiche

Für die Textausgabe läßt sich zusätzlich der Bildschirm in eine obere und untere Hälfte splitten. Hier ist jeweils nur eine der beiden Hälften für die Textausgabe aktiv und auch nur dieser Bereich scrollt:

<CTRL CBM X> - ←m,1

... erlaubt Textausgabe nur für die obere Hälfte des Bildschirms. Bei Multicolor wird die untere Hälfte auf Multicolor geschaltet.

<CTRL CBM C> - ←m,2

... danach läßt sich nur die untere Hälfte für die Textausgabe nutzen. Bei Multicolor ist nur die obere Hälfte im Multi-Mode.

<CTRL CBM Z> - ←m,0

... schaltet um auf Textausgabe und -scrolling für den gesamten Bildschirm. Multicolor wirkt über den gesamten Bildschirm.

Lösch- und Scrollfunktionen

Sechs unterschiedliche Arten zum Bildschirmlöschen oder Zeilen- und Spaltenverschieben machen die Textausgabe am Bildschirm komfortabel:

<CTRL W> - CHR$(23)

... löscht die Zeile, in der sich der Cursor momentan befindet.

<CTRL G> - CHR$(7)

... scrollt den Bildschirm eine Zeile nach oben. Die Anzahl der gescrollten Bildschirmlinien ist dabei abhängig vom momentan eingestellten Auflösungsmodus.

<CTRL B> - CHR$(2)

... scrollt den Bildschirm eine Zeile nach unten. Auch hier ist die Anzahl der Bildschirmlinien vom momentan eingestellten Auflösungsmodus abhängig.

<CTRL CBM V> - ←m,6

... scrollt die Zeile, in der der Cursor steht, um acht Pixel nach links. Es wird grundsätzlich um acht Pixel verschoben, unabhängig vom Modus.

<CTRL CBM B> - ←m,7

... scrollt die Zeile, in der der Cursor steht, um acht Pixel nach rechts. Verwenden Sie im Programm den Befehl »←m 7«. Auch hier wird grundsätzlich um acht Pixel verschoben, unabhängig vom Modus.

<CBM>

... stoppt das Scrollen des Bildschirms. Dieser Code kann nicht in ein Programm geschrieben werden.

Farben im Interlace

Im Gegensatz zum normalen Betriebssystem des C64 läßt sich bei einfachen Farben der Bildschirm wieder teilen. Die Hintergrundfarbe kann damit für die obere und untere Hälfte getrennt bestimmt werden. Wenn Sie die Farben bestimmen wollen, sollten Sie in folgender Reihenfolge vorgehen:

1. Überlegen Sie sich, ob Normalfarben oder Multicolor verwendet werden sollen. Achten Sie bei Multicolor auf den Splitscreen (s.o.). Die Darstellung in Multicolor liegt gegenüber dem Eingabe-Editor immer in der anderen Bildschirmhälfte.
2. Bildschirmhälfte oder Gesamtbildschirm zur Farbsetzung bestimmen (s.u.).
3. Den zu ändernden Bereich auswählen (Rand, Multi-Farbe 0, Multi 1 = Zeichenfarbe, Multi 2 = Hintergrundfarbe oder Multi3).
4. Erst jetzt die Farbe ändern

Multicolor

<CTRL C> - ←m,9 oder ←m,8

... schaltet Multicolor ein bzw. aus. <CTRL C> hat dabei Schalterfunktion und wechselt zwischen Ein und Aus bei Direkteingaben. ←m,9 schaltet im Programm Multicolor ein, ←m,8 schaltet aus.

Bildschirmhälfte

<CTRL CBM P> - ←m,4

... bestimmt die obere Bildschirmhälfte für Farbsetzfunktionen (außer Rahmen).

<CTRL CBM @> - ←m,5

... bestimmt die untere Bildschirmhälfte für Farbsetzfunktionen (außer Rahmen).

<CTRL CBM O> - ←m,3

... der gesamte Bildschirm ist für Farbsetzfunktionen vorgewählt (außer Rahmen).

Bereichsvorwahl

<CTRL O> - CHR$(15)

... wählt den Rahmen als nächste einzustellende Farbe.

<CTRL P> - CHR$(16)

... wählt Multicolor-Farbe 0 als nächste einzustellende Farbe.

<CTRL K> - CHR$(11)

... wählt Multicolor-Farbe 1 als nächste einzustellende Farbe. Diese entspricht der Zeichenfarbe im Normalmodus.

<CTRL L> - CHR$(12)

... wählt Multicolor-Farbe 2 als nächste einzustellende Farbe. Sie entspricht der Hintergrundfarbe im Normalmodus.

<CTRL :> - CHR$(27)

... wählt Multicolor-Farbe 3 als nächste einzustellende Farbe.

Farbeneinstellung

<CTRL 1> bis <CTRL 8>

... entspricht den Farbnummern »0« bis »7« und funktioniert wie auf der Tastatur bezeichnet.

<CBM 1> bis <CBM 8>

... entspricht den Farbnummern »8« bis »15« und funktioniert ebenfalls wie auf der Tastatur bezeichnet.

Interlacefarben

←k,Adresse,Farbe

... setzt die Farbe des Zeichens (0 bis 255) an die »Adresse« 0 bis 999. Dabei ist der Bildschirm von links oben nach rechts unten in Zeilen eingeteilt, in denen jeweils 8 x 8 Pixel große Felder gesetzt werden. »Farbe« bezeichnet eine Mischung aus zwei der 15 Grundfarben des C64 (s. Handbuch). Sie werden im Interlace jede 1/25 Sekunde getauscht. Dem Auge entsteht dadurch der Eindruck einer Mischfarbe. Die Farben berechnen sich: Farbe1 + (Farbe2 * 16).

Beispiel: Sie möchten Rot und Braun mischen. Laut Handbuch hat Rot die Farbnummer »2« und Braun »9«. Die Berechnung lautet: 2+(9x16)=146 oder 9+(2x16)=41. Beide Werte ergeben die gleiche Mischfarbe. Beachten Sie, daß hier nur etwas sichtbar wird, wenn an der entsprechenden Position tatsächlich auch ein Zeichen oder eine Grafik am Bildschirm ist.

`←k,0,41`

setzt an die oberste linke Bildschirmposition die oben berechnete Mischfarbe. Sichtbar wird sie, wenn Sie beispielsweise mit dem Cursor diese Position anfahren und den Buchstaben »A« eintippen. Die beiden Farbspeicher liegen:

1. von $8C00 bis $8FE7 und

2. von $DC00 bis DFE7.

←l,Adresse,Farbe

... wie oben, nur wird die Hintergrundfarbe gesetzt. »Adresse« ist wieder die Position, an die die Mischfarbe »Farbe« geschaltet wird.

←p,Adresse,Byte

... POKEd den Wert von »Byte« (0 bis 255) an die Speicherposition »Adresse« (0 bis 65535).

←s

... scrollt den Mischfarbenschirm (ohne Zeichen oder Grafik) um acht Pixel nach oben.

Grafikfunktionen

Da doppelte Auflösung zur Verfügung steht, lassen sich zwei Hiresbilder auf einen Bildschirm zaubern. Zwei Modi stehen dafür zur Verfügung. Entweder Sie setzen zwei Screens nebenan oder übereinander. Eine gewisse Verzerrung ist dabei leider nicht auszuschließen. Beim Kombinieren zweier Bilder bietet es sich an, zuerst ein Hiresbild in den Arbeitsspeicher zu laden (<CBM F1>) und danach in die Interlace-Darstellung zu transferieren (<F1> bis <F4>). Danach laden Sie das andere Bild in den Arbeitsspeicher (<CBM F1>) und schieben es in den noch freien Bereich des Schirms. Anschließend speichern Sie das Interlace-Bild (<CBM F7>). Beim »saven« wird die Interlace-Darstellung gepackt und bekommt ein Maschinenprogramm an den Anfang. Wenn Sie es später außerhalb von Interlace 64 (mit »LOAD"Name",8«) laden und mit RUN starten, zeigt diese Routine das Bild.

<F3>

... schiebt ein Hiresbild aus dem Arbeitsspeicher in die linke Hälfte des Interlace-Screens.

<F4>

... schiebt ein Hiresbild aus dem Arbeitsspeicher in die rechte Hälfte des Interlace-Screens.

<CTRL A> - CHR$(1)

... convertiert die obere Hälfte der Interlace-Darstellung in Hires im Arbeitsspeicher.

<CTRL Z> - CHR$(26)

... convertiert die untere Hälfte der Interlace-Darstellung in Hires im Arbeitsspeicher.

<CTRL F> - CHR$(6)

... convertiert die linke Hälfte der Interlace-Darstellung in Hires im Arbeitsspeicher.

<CTRL V> - CHR$(22)

... convertiert die rechte Hälfte der Interlace-Darstellung in Hires im Arbeitsspeicher.

<F5>

... splittet die Interlacedarstellung und zeigt oben die geraden und unten die ungeraden Zeilen.

<F6>

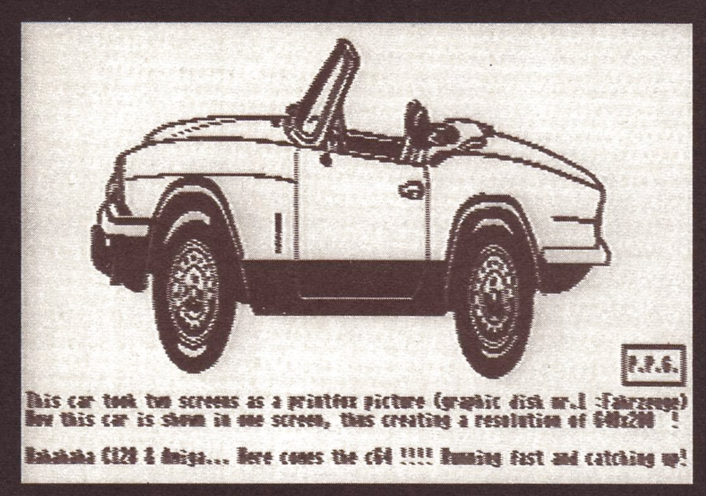
... splittet die Interlacedarstellung und zeigt links die ungeraden und rechts die geraden Bits.

<F7>

... schaltet um auf Hires-Darstellung.

<F6>

... zeigt ein Interlace-Bild vom Arbeitsspeicher. Vorher muß allerdings eins gespeichert oder geladen sein. Achtung: Der Befehl löscht den Basic-Speicher.



[2] Normalerweise nicht möglich: Zwei HIRES nebeneinander



[3] Zwei Bilder lassen sich auch untereinander anordnen, allerdings wirkt das Ergebnis gestaucht

Lade- und Speicherfunktionen

<CBM F1> - ←h,"Name",8

... lädt ein HIRESbild in den Arbeitsspeicher. Bei <CBM F1> folgt eine Frage nach dem Programmnamen, bei »←h,"Name",8« geben Sie ihn für »Name« an. »,8« lädt von Diskette.

<CBM F5> - ←m,10

... lädt ein Interlacebild nach Eingabe des Namens.

<CBM F7> - ←m,11

... speichert ein Interlacebild nach Eingabe des Namens.

Umwandlungsbefehle für Grafiken

<F1>

... schiebt ein Hiresbild aus dem Arbeitsspeicher in die obere Hälfte des Interlace-Screens.

<F2>

... schiebt ein Hiresbild aus dem Arbeitsspeicher in die untere Hälfte des Interlace-Screens.

Zeichensatzfunktionen

<CBM F3> - ←c,"Name",8

... lädt einen Zeichensatz von Diskette. Bei »←c,"Name",8« geben Sie ihn für »Name« an. »,8« lädt von Diskette.

Hinweis: Die gespeicherten Interlace-Bilder werden mit (LOAD"Name",8) und RUN ohne Programm sichtbar. (gr)

Kurzinfo: Interlace 64

Programmart: Grafikerweiterung
Laden: LOAD"Interlace 64",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Nach <RUN/STOP RESTORE> muß mit SYS49152 wieder gestartet werden
Benötigte Blocks: 17
Programmautor: Paul Guldenaar

Blitter – 3-D-Movie

GEOMETRISCHES HEIMKINO

Mit »Blitter« entwerfen Sie dreidimensionale Grafikkörper und lassen Sie wie auf der Filmleinwand übers Bild huschen.

Hires-Bilder sind zwar schön anzusehen, aber mit der Zeit wird's langweilig, es rührt sich halt nichts! Um Bewegung in die Grafik zu bringen, bleiben lediglich Sprites, denn: programmierte Hires-Bilder werden in Basic zu langsam gezeichnet. Um Bewegungssequenzen zu erzeugen, braucht man viele Hires-Bildschirme zu jeweils 8000 Byte (jeder mit geringfügigen Änderungen): Dazu ist der Speicher des C64 zu knapp.

Doch es geht auch anders: Man berechnet nicht die Fläche der Körper, sondern arbeitet mit 3-D-Drahtmodellen. Sie setzen sich nur aus Koordinaten für die Eckpunkte und Linien zusammen. Ein Computerprogramm berechnet die äußeren Positionen eines oder mehrerer Körper. Alle Daten und Koordinaten werden in einer Tabelle vermerkt. Anschließend zeichnet der C64 eine Filmsequenz und zeigt die berechneten Linien auf dem Schirm, ausgehend von den entsprechenden Eckpunkten. Der Animationsteil des Programms verwendet die Koordinatentabelle und läßt die vorher berechneten

Kurzinfo: Blitter

Programmart: Grafik-Animationsprogramm
Laden: LOAD "BLITTER",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: berechnet Bewegungssequenzen Schritt für Schritt und legt Grafikkoordinaten in gesonderter Tabelle ab
Benötigte Blocks: 16
Programmautor: Oliver Strunk

Sequenzen hintereinander quasi als Film ablaufen. Nichts anderes macht »Blitter«. Die Basic-Erweiterung setzt dazu eine spezielle LINE-Routine ein.

Ein paar Vergleichsdaten: Die LINE-Routine von Simon's Basic zeichnet ca. 4000 Pixel in der Sekunde, sehr schnelle Grafik-Erweiterungen schaffen ungefähr 12 000. »Blitter« aber bringt's auf 30 000 Bildpunkte pro Sekunde. Für Autofahrer: Das entspricht einer Geschwindigkeit von 80 km/h. Handelt es sich um Geraden, erreicht das Programm sogar 180 000 Punkte pro Sekunde (500 km/h). Dabei geht die meiste Zeit gar nicht beim Zeichnen einer Linie drauf, sondern beim Bildschirmlöschen nach jeder Sequenz. Das geschieht 17mal pro Sekunde, ohne daß überhaupt eine einzige Linie gezeichnet wurde.

»Blitter« ist keine der üblichen Basic-Erweiterungen des C64 zur Grafikprogrammierung (dazu fehlen wichtige Befehle, wie BOX, CIRCLE, FILL usw.), sondern ein Animationsprogramm: Es soll Filmschritte berechnen, die sich z.B. als Vorspänne, Grafikdemos oder kurze Werbefilme verwenden lassen. Außerdem kann man Endlosfilme gestalten, wenn man zyklische Vorgänge (z.B. einen Viertaktmotor) als Vorlage nimmt. Oder: Denken Sie an einen rotierenden Globus.

Laden Sie das Grafikprogramm mit:

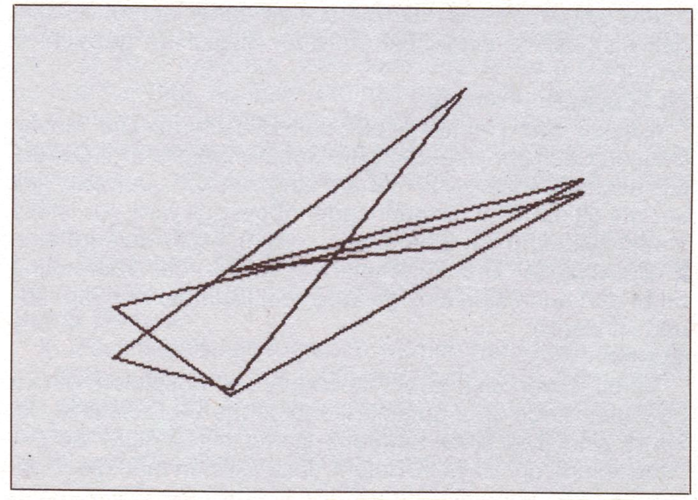
```
LOAD "BLITTER",8
```

und starten Sie es mit RUN. Es verändert sich nichts auf dem Bildschirm: Wenn sich der Cursor wieder meldet, ist die Befehlserweiterung initialisiert. Nach einem Reset kann man das Tool mit

```
SYS 19712
```

ebenfalls wieder aktivieren.

Wem der Begriff »Blitter« bekannt vorkommt, denkt wahrscheinlich an den Amiga. Dort fungiert dieser Teil des Grafikchips »Agnus« als Zwischenspeicher für Animationssequenzen, die er in blitzartiger Geschwindigkeit auf den Bildschirm bringt und damit einen reibungslosen Bewegungsablauf simuliert. Das verwirklicht unser Programm »Blitter« auch auf dem C64 – selbstverständlich mit allen Einschränkungen, die den C64 vom Amiga unterscheiden. Der Atari Mega-STE besitzt einen separaten Baustein, der ebenfalls »Blitter« heißt – mit exakt denselben Funktionen.



[1] Eine Zufallsroutine bestimmt das Aussehen der dreieckigen Körper

Unsere Software-Erweiterung »Blitter« für den C64 verwendet zwei wichtige Systemvariablen: **datastart** und **dataend**. Sie fungieren als Zeiger für die Datentabelle im Speicher und werden vom Programm automatisch verwaltet: datastart liegt immer bei $8000 (32768), dataend wird als Low-/Highbyte in den Adressen $03FE/$03FF (1022/1023) abgelegt (s. Tabelle).

Jetzt können Sie eigene Basic-Programme für bewegte Grafiksequenzen entwickeln. Dazu stehen Ihnen folgende neue Befehle zur Verfügung:

START

schaltet den Hires-Bildschirm ein und löscht ihn.

CLEAR

löscht den gesamten Datenspeicher: Dabei wird der Zeiger von dataend auf den Wert von datastart gesetzt.

SPEICHER wert

setzt die Zeiger für datastart. Da nur das Highbyte aktiviert wird (das Lowbyte ist immer »0«), sind für »wert« Zahlen zwischen »0« und »255« möglich. Allerdings: Sinnvoll sind nur Nummern über »128« (=$8000), sonst wird »Blitter« überschrieben. Der freie Datenspeicher reicht maximal bis $CF05 (52997).

NRML

schaltet von Hires-Grafik wieder zum Textbildschirm. Diese Anweisung dürfen Sie auch eintippen, wenn der Grafikbildschirm noch aktiv ist. Nach <RETURN> wird die Anweisung trotzdem übernommen.

LINE x1,y1,x2,y2

zeichnet eine Linie und trägt die Koordinaten in der Datentabelle ein. Fünf Bytes werden dafür verbraucht. »x1, y1« sind die Startkoordinaten (horizontal und vertikal), »x2,y2« die End-

punkte. Diese vier Parameter dürfen Werte zwischen - 32768 bis 32767 annehmen. Achtung: Das gilt nicht die für Hires-Grafiken übliche Bildschirmeinteilung! Der Ursprung (0,0) liegt in der Bildmitte - das sind eigentlich die Koordinaten 160,100. Als oberer Rand gilt in y-Richtung (vertikal) die Zahl - 99, der untere Rand liegt bei y = 99. Der rechte Bildrand hat die horizontale Koordinate x = 158, der linke x = - 158. Schnittkoordinaten mit den Bildschirmrändern berechnet »Blitter« automatisch.

BILD

»Blitter« verwaltet zwei Hires-Bildschirme: Der eine liegt bei $6000 (24576), der andere unterm ROM bei $E000 (57344). BILD schaltet auf den zweiten um und löscht dabei den ersten. In der Datentabelle wird das ebenfalls registriert. Zwei Vermerke hintereinander zeigen »Blitter«, daß eine Filmsequenz abgeschlossen ist. Sie sollten daher darauf achten, daß mindestens eine Linie auf dem Bildschirm gezeichnet wird.

FILM speed + anzahl

aktiviert das Heimkino auf den Bildschirm: Die Movie-Sequenz läuft ab. »speed« ist die Geschwindigkeit: »1« bedeutet »höchste Stufe«, »127« ist am langsamsten. »anzahl« gibt an, wie oft der Film hintereinander abgespult wird: »0« ist die Voreinstellung (Film läuft einmal durch), »128« erzeugt eine Endlosschleife. Die Parameterzahl hinter der Anweisung FILM darf auch eine einzige Zahl sein (Summe aus »speed« und »anzahl«).

BLOCK

Damit lassen sich zwei Filmsequenzen abspielen, die im Speicher durch zwei aufeinanderfolgende BILD-Befehle getrennt sind. Die erste Sequenz startet mit <FILM wert>, dann kommt der BLOCK-Befehl. Jetzt aktiviert man die nächste Sequenz ebenfalls mit <FILM wert> usw. Will man anschließend die Datentabelle speichern, muß man per SPEICHER zunächst den Wert für datastart wieder auf den Beginn der gesamten Filmsequenz zurückstellen.

WRITE "name"

speichert eine Datentabelle von datastart bis dataend auf Diskette.

FETCH "name"

lädt die Datentabelle wieder in den Computerspeicher.

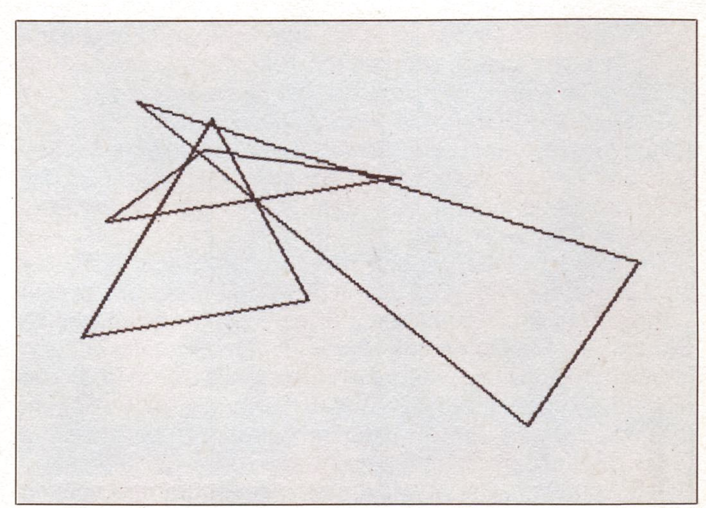
Eigene Filme entwerfen

Die zusätzlichen Basic-Befehle von »Blitter« dienen nur einem Ziel: einen Film aus berechneten und abgelegten Daten herzustellen.

Per Basic-Programm definiert man einen geometrischen Körper, z.B. durch eine mathematische Formel oder festgelegte Eck-Koordinaten. Die Werte für x und y legt man am besten in Datazeilen ab. Das Programm soll nun die Eckpunkte für die Sequenzen berechnen. Trotz des relativ geringen Speicherplatzes darf man getrost auch Datenfelder und Variablen zur internen Datenspeicherung verwenden: Dann läßt sich in der Berechnungsroutine immer auf bestimmte Werte zurückgreifen. Jetzt veranlaßt das Programm die LINE-Anweisung, einen Teil des Körpers zu zeichnen und die Werte in der Datentabelle zu registrieren. Es gibt keinen anderen Weg, die benötigten Zahlen in der bewußten Tabelle zu verewigen.

Nun kommt der Bewegungseffekt: Die verschiedenen Phasen trennt man jetzt durchs BILD-Kommando. Hat man viele Linien nebeneinander gezeichnet, erhält man ohne diese Trennung eine ausgefüllte Fläche, denn alles erscheint jetzt auf **einem** Bildschirm. Erst durch die Separation mit BILD (vor dem Zeichnen der nächsten Linie), entsteht die Illusion eines wandernden Strichs. In der Praxis bedeutet das: Man muß jedes Einzelbild berechnen lassen. Das erledigt man am besten mit einer FOR-NEXT-Schleife. Dazu benötigt der erste Durchlauf einige Zeit: Er dient der Berechnung und dem Erfassen der Koordinatenwerte für die Datentabelle. Alles läuft quasi in Zeitlupe ab. Ist das vorbei, starten Sie die Animation mit FILM: Der Ablauf wirkt jetzt fließend. Damit lassen sich Animationssequenzen, deren Berechnung oft Stunden oder gar Tage dauern kann (je nach programmierter Rechenformel), innerhalb weniger Sekunden zu einem Film verschmelzen. Hier erweist sich die Möglichkeit zur Speicherung der gesamten Datentabelle als sehr nützlich: Komplexe und langwierige Berechnungen müssen nur einmal gemacht werden.

»Blitter« reserviert exakt 20 229 Bytes für die Datentabelle. Sie liegt zwischen den beiden Hires-Bereichen, die außerdem noch jeweils 1000 Byte fürs Farb-RAM abzwacken. Mindestens zwei Grafikbildschirme sind unbedingt nötig, um ein flimmerfreies Bild zu bekommen. Bei nur einem Schirm müßte man sich damit abfinden, daß er ständig gelöscht und Pixel für Pixel jedes Einzelbild erneut aufgebaut würde. Das macht »Blitter« verdeckt in einem dafür reservierten zweiten Speicherbereich, während der erste die Resultate zeigt. Das Programm schaltet erst dann um, wenn die Grafik fertig ist. Da jede Linie fünf Bytes und jeder Bildschirmwechsel ein weiteres in der Tabelle belegt, passen maximal 3900 Linien in den Datentabellen-Speicher. Verwendet man pro Bild z.B. 30 Linien mit einer durchschnittlichen Länge von 50 Pixeln, ergibt das 130 Bilder. Sie werden mit einer Frequenz von neun Grafikschirmen pro Sekunde in einem 14 s dauernden



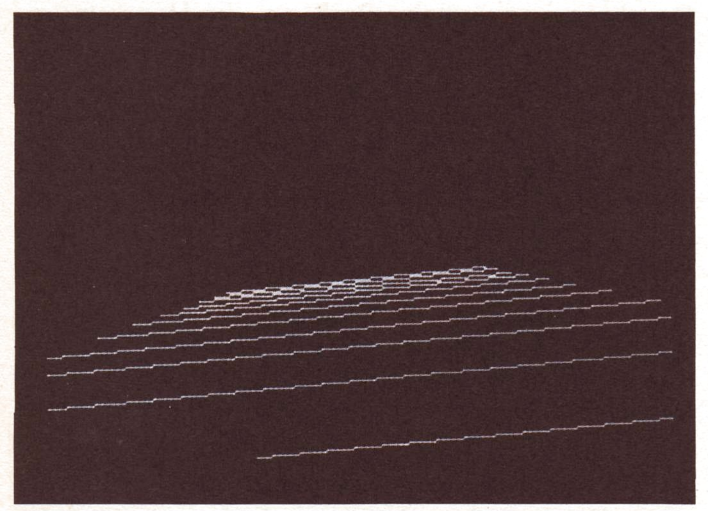
[2] Sämtliche markanten Eckpunkte werden in der Datentabelle gespeichert

Speicheraufteilung »Blitter«

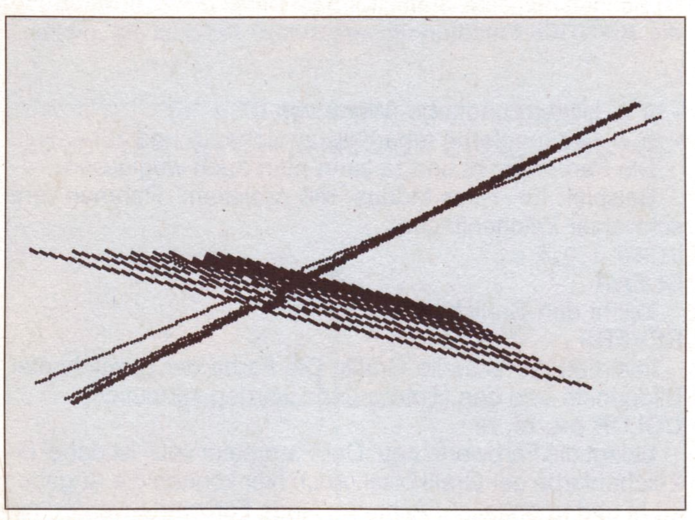
Bereich	Funktion
$0000 bis $035F	Zeropage, Stack
$035F bis $03FF	temporärer Speicher
$03FE / $03FF	Pointer auf Ende der Datentabelle (dataend)
$0400 bis $07F7	Video-RAM
$0801 bis $4CFF	Basic-Speicher (17 KByte)
$4D00 bis $5C00	Erweiterung »Blitter«
$5C00 bis $5FF7	Farb-RAM 1
$6000 bis $7F3F	Hires-Bildschirm 1
$8000 bis $CF05	Datentabelle
$D000 bis $DBFF	unbenutztes RAM unterm ROM
$DC00 bis $DFF7	Farb-RAM 2
$E000 bis $FF40	Hires-Bildschirm 2

Film ausgegeben. Der Erfahrungswert für die Bildwiederholfrequenz, mit der ein Film fließend abläuft: Zehn Bilder pro Sekunde. Bei weniger als fünf Bildern wird's allerdings kritisch.

»Blitter« benutzt eine 320 x 200-Grafikpixel-Auflösung. Damit bleiben auch kleine Strukturen noch gut sichtbar. Das Programm berücksichtigt nur die zu Beginn Ihres Animationsprogramms eingestellten Vordergrund- und Hintergrundfarben: der Multicolormodus wird nicht akzeptiert. Wenn sich nämlich zwei verschiedenfarbige Linien schneiden, erscheint der Schnittpunkt in der dritten Farbe. Solche Situationen abzufragen, kostet Einbußen bei der schnellen Rechenzeit des Programms oder ein undurchschaubares Farbchaos in Kauf nehmen.



[3] Wie durchs Cockpit-Fenster betrachtet: Blitter Demo 2.



[4] Viele Variationsmöglichkeiten stehen zur Verfügung

Wenn Sie zwei Datentabellen miteinander verbinden möchten, müssen Sie folgende Anweisungen verwenden (im Programm oder Direktmodus):

```
SPEICHER
CLEAR
FETCH "(Dateiname 1)"
FETCH "(Dateiname 2)"
WRITE "(Datei verbunden)"
```

Dabei nutzt man die Funktion des Programms aus, neue Daten immer hinter den bestehenden anzuhängen. Wenn also Berechnungen zu lange dauern, ist es kein Problem, an gewünschter Stelle abzubrechen. Speichern Sie jetzt den unvollständigen Datensatz mit markantem Namen. Die Tabelle kann zu einem anderen Zeitpunkt wieder geladen werden.

Jetzt lassen sich die restlichen Koordinaten berechnen. Ist das Programm endlich fertig, wird der gesamte Datensatz unter anderem Dateinamen auf Diskette gesichert. Achtung: »Blitter« speichert keine Hires-Bilder mit 33 Blocks auf Diskette!

»Blitter« ist ein sehr variables Tool, das vom Benutzer im entsprechenden Basic-Programm allerdings exakte Koordinaten (in Datas abgelegt) oder hieb- und stichfeste Rechenformeln benötigt. Zwei Demoprogramme auf dieser Sonderheft-Diskette zeigen deutlich, wie man die neuen Basic-Befehle effektvoll einsetzen kann. Beide Demos lassen sich jederzeit ändern und ausbauen:

Blitter Demo1

Sehen Sie sich nach dem Laden das Listing an. Markante Programmzeilen sind mit REM-Erläuterungen dokumentiert. Die Bildschirmfarben werden in Zeile 9 eingestellt, ab Zeile 25 definiert der Autor drei Dreiecke. Ihre geometrische Form berechnet das Programm nach dem Zufallsprinzip (Zeilen 70 und 90 bis 150). Ab Zeile 170 hält das Demo die einzelnen Animationssequenzen fest (Abb. 1 und 2), deren Werte in der Datentabelle festgehalten werden. Zeile 220 startet den Film: »speed« = 1 (Höchstgeschwindigkeit), »anzahl« = 128 (endlos).

Mit <RUN/STOP RESTORE> kann man die Filmvorführung abbrechen. Der Computer befindet sich wieder im Direktmodus. Das Listing kann editiert werden.

Blitter Demo2

In diesem Beispielprogramm werden nur Linien gezeichnet (Subroutine ab Zeile 2000) und ebenfalls in Tabellenwerten abgelegt. Die Animationssequenz wird wieder in einem Endlosfilm gezeigt (Abb. 3). Erhöhen Sie die Koordinaten y1, z1, y2 und z2, definieren Sie zwei weitere LINE-Anweisungen und fügen diese als Zeilen 2080 und 2090 an. Dazu muß man in Zeile 2070 den Befehl RETURN löschen und in Zeile 3000 eintragen. Das Demoprogramm ist um eine Variante reicher!

Hinweise für Programmierer

»Blitter« belegt unmittelbar nach dem Laden den Speicherbereich von $0801 (2049) bis $173F (5951). Nach RUN wird die Hauptroutine des Programms im Speicher nach $4D00 (19712) verlegt. Die Zeiger der Betriebssystemroutine zur Interpretation der eingegebenen Basic-Befehle ($0308/$0309) sind jetzt auf die Adresse $4D6C (19820) gerichtet, in der die neuen Anweisungen von »Blitter« (z.B. SPEICHER, FETCH, FILM usw.) oder normale Befehle des Basic 2.0 erkannt werden. Dabei wird auf die Systemroutinen CHRGET ($0073) und CHRGOT ($0079) zurückgegriffen. Dann verzweigt »Blitter« zu den entsprechenden Unterprogrammen, die diese Befehle ausführen. Die neuen Befehle findet man im Klartext ab $4D19 (19737) bis $4D5D (19805).

Zwei Unterprogramme muß »Blitter« ständig anspringen, um die Speicherstelle $01 für seine Zwecke zu manipulieren (Basic- und Kernel-ROM aus-, bzw. wieder einschalten): $4DEF (19951) und $4DF5 (19957). Die Speicherroutine zum Sichern der Datentabellen beginnt bei $4E72 (20082), das Unterprogramm zum Laden solcher Dateien finden Sie ab Adresse $4E3D (20029). Zur Ablage temporärer Berechnungsdaten werden eine Menge Adressen des Kassettenpuffers genutzt – »Blitter« ist daher nicht mit einer Datasette lauffähig. Sämtliche Lade- und Speicherroutinen (WRITE, FETCH) wirken nur mit einer Diskettenstation.

Man muß schon ein wenig überlegen und experimentieren, bis grafische Körper berechnet und deren Ausgangswerte definiert sind: Das Programm nimmt Ihnen aber die weitaus umfangreichere Arbeit zum Berechnen der Animationsdaten ab und startet das Computer-Heimkino automatisch!

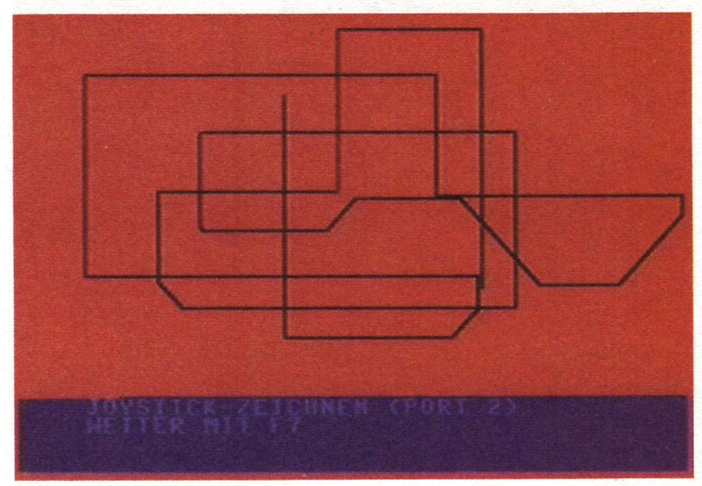
(Oliver Strunk/bl)

Turtle Graphics

- Grafik im Vorbeigehen

SCHILDKRÖTE ALS SPRINTER

Kürzer geht's kaum: Dennoch erzeugt der spartanische Befehlssatz der Basic-Erweiterung »Turtle Graphics« vorzügliche Grafiken – im Handumdrehen!



[1] Schnell mal eine Grafik auf dem Bildschirm entwerfen: die JOYSTICK-Funktion

Es gibt viele großartige Erweiterungen zum Basic 2.0 des C 64, die eine Unmenge neuer Befehle zur Verfügung stellen, um hochauflösende Grafiken zu entwickeln. Wer jedoch nur mal schnell ein Bild mit wenigen Pinselstrichen erzeugen möchte, wird von den langwierigen Lade- und Installationsorgien dieser komplexen Grafikpakete bald genervt. Ganz davon abgesehen, daß man umfangreiche Beschreibungen und Handbücher wälzen muß, nur um den Befehl zum Ziehen einer Linie zu finden.

Die 16 Befehle von »Turtle-Grafik« dagegen kann man sich leicht merken. Noch ein Vorteil dieser Basic-Erweiterung: Es lassen sich Programme entwickeln, die eine Grafik auf dem Bildschirm fabrizieren. Oder: Der Joystick in Port 2 fungiert als Zeichenpinsel. Jede Grafik läßt sich speichern und später wieder in den Computer holen.

Laden Sie das Programm mit:

```
LOAD "TURTLE GRAPHICS",8
```

und starten Sie es mit RUN.

Wenn der Einschaltbildschirm erscheint, ist die Erweiterung so lange aktiv, bis der Resetknopf gedrückt oder der C 64 ausgeschaltet wird.

Der Textbildschirm wird eingeschaltet. Per <F1>-Taste schaltet man zwischen Textmodus und hochauflösender Grafik um. Auf dem Hires-Bildschirm erkennen Sie in der Mitte ein blinkendes Pünktchen: Das ist die Schildkröte (Turtle), sprich der Grafikcursor. Normalerweise ist der nämlich unsichtbar.

Die beiden Betriebsmodi stehen sofort zur Verfügung:

Freihand-Zeichnen per Joystick oder Lightpen

Diesen Modus muß man mit einem Befehl im Textbildschirm einleiten:

JOYSTICK verzögerung: Wenn verzögerung = 0, malt das Programm am schnellsten. »255« ist die geringste Geschwindigkeit. Den Ideal-Speed sollte jeder selbst testen, wir empfehlen:

```
JOYSTICK 100
```

Der Cursor verschwindet auf dem Textbildschirm: Jetzt muß man mit <F1> in den Grafikmodus schalten. Mit dem Joystick bewegt man den Grafikpunkt in jede beliebige Richtung. Wenn Sie den Feuerknopf drücken, werden alle Pixel entlang der Zeichenlinie gesetzt (Abb. 1). Sollen Bildpunkte gelöscht werden, muß dies vorher mit dem MODE-Befehl eingestellt werden (s. Beschreibung). Ist Ihre Zeichnung fertig, schalten Sie mit der <F1>-Taste wieder zum Textbildschirm. Damit der Cursor für weitere Befehlseingaben erscheint (um z.B. das Bild auf Diskette zu speichern), müssen Sie <F7> drücken.

Wer einen Lichtgriffel (Lightpen) besitzt, kann den ebenfalls (neben dem Joystick) als direktes Zeichengerät einsetzen. Dieses spezielle Eingabewerkzeug muß im Joystickport 1 angeschlossen sein. Mit folgender Anweisung wird der Lightpen aktiviert:

```
LPEN
```

Grafikbefehle auf einen Blick

Ansonsten gelten die gleichen Regeln wie bei der JOYSTICK-Anweisung. Zum Zeichnen müssen Sie statt des Feuerknopfs die CTRL-Taste bzw. den entsprechenden Knopf am Lichtgriffel drücken. Diesen Modus verläßt man ebenfalls mit <F7>.

Grafiken per Basic-Programm

Selbstverständlich lassen sich exakte Bilder mit geplanten Koordinatenangaben erzeugen – der Programmieraufwand ist geringer, als Sie denken.

Folgende neuen Basic-Befehle stehen zur Verfügung:

HIRES md, hi, ra

- md = Modus: 1 = Hires-Bildschirm ein, 0 = aus,
- hi = Hintergrundfarbe (Werte von 0 bis 15),
- ra = Rahmenfarbe (ebenfalls zwischen 0 und 15).

Die Parameter hi und ra kann man auch weglassen.

Beispiel für Hires-Modus mit violettem Rahmen und schwarzer Zeichenfläche:

```
HIRES 1,0,4
```

CLEAR

löscht den Grafikbildschirm.

REVERS

invertiert die aktuelle Grafik: Die Farbe der gezeichneten Bildpunkte und des Hintergrunds werden vertauscht.

COLOR pu, hi, ra

belegt die Farbwerte neu. Der Parameter »pu« ist dabei die Zeichenfarbe der Grafikpixel. Auch hier können die Angaben zu hi und ra entfallen. Achtung: Neue Farbwerte werden nur übernommen, wenn der Grafikbildschirm nach der COLOR-Anweisung mit CLEAR gelöscht wird.

GSAVE "Bildname",8

speichert den aktuellen Grafikbildschirm unter einem beliebigen Namen auf Diskette. Die Datei belegt 33 Blöcke.

GLOAD "Bildname",8

lädt die Grafik wieder, die sich nun verändern oder weiter bearbeiten läßt.

DEG ri

bestimmt die Bewegungsrichtung des Grafikcursors: Acht Werte (zwischen 0 und 7) sind möglich (Abb. 3).

MOVE x

bewegt den Grafikcursor um x Punkte in der Richtung DEG ri. Nach der HIRES-Anweisung steht er automatisch in der Bildschirmmitte (x = 160, y = 100). Der Parameter ri wurde auf »0« gesetzt, der MODUS ist »0« (s. Beschreibung).

Die beiden letztgenannten Basic-Befehle simulieren quasi die Joystickführung und den Druck auf den Feuerknopf beim Freihandzeichnen.

LTURN x

dreht den Grafikcursor um x Bildpunkte nach links,

RTURN x

das gleiche, aber nach rechts. »x« darf Werte bis maximal »255« annehmen. Beispiele:

```
LTURN 50 (50 Pixel nach links)
RTURN 20 (20 Bildpunkte rechts)
```

PLOT x, y

plaziert die »Turtle« (Grafikcursor) an gewünschter Position auf dem Hires-Bildschirm. Die linke obere Ecke wird durch die Werte »x =0, y = 0« definiert, die rechten unteren Koordinaten liegen bei »319,199«.

MODE m

Diese Anweisung hängt unmittelbar mit dem MOVE-Befehl zusammen. Die Zahl für »m« bestimmt, wie die Grafikpixel auf dem Bildschirm erscheinen.
- 0: Punkt setzen,
- 1: Punkt löschen,
- 2: inverses Pixel,
- 3: nichts verändern.

Vor allem, wenn Sie bereits fertige Grafiken ergänzen möchten, müssen Sie in Ihrem Programm auf den Modus 3 zurückgreifen.

»Turtle Graphics« erzeugt auf Wunsch auch einen Splitscreen (in Hires- und Textmodus geteilter Bildschirm): Vier Textzeilen werden unterhalb der Grafik eingeblendet. Im Direktmodus erledigt das die Taste <F3>, erneuter Druck zaubert das Textfenster wieder weg.

Die Basic-Befehle dazu:

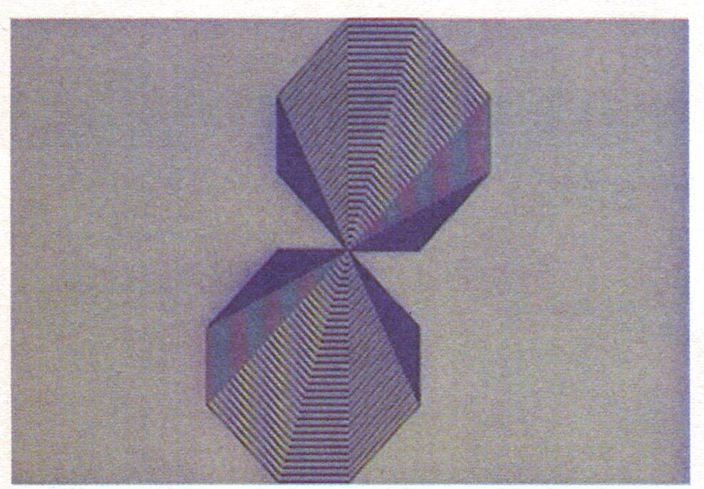
WINDOW 1: schaltet den vierzeiligen Textbereich ein (um z.B. Erläuterung zum Grafikbildschirm unterzubringen),

WINDOW 0: blendet das Textfenster wieder aus.

Im Direktmodus bringt die <F5>-Taste den Textcursor in die linke obere Ecke dieses Textbereichs, den man aber per <CRSR aufwärts> trotzdem verlassen kann. Der blinkende Cursor ist dann allerdings unsichtbar, da dieser Bildschirmbereich zur hochauflösenden Grafik gehört: Mit <CRSR abwärts> können Sie den Ausreißer wieder zurückholen. Möchten Sie die <F5>-Funktion im Basic-Programm erzeugen, lautet die Anweisung:

```
PRINT CHR$(135)
```

Diese Funktion wirkt wie die HOME-Taste im normalen C-64-Modus und setzt den Cursor in die linke obere Bildschirmecke des Textfensters. Man kann den Cursor allerdings nicht über diese Grenzen hinaus bewegen.



[2] Leicht verständliche Befehle erzeugen attraktive Grafiken, die sich mit GSAVE speichern lassen

JOYSTICK-RICHTUNGSWERTE DER TURTLE

2 / 3 / 1 / 4 / 0 / 5 / 7 / 6

BEFEHL: DEG WERT

[3] Diese Joystick-Richtungen akzeptiert der DEG-Befehl

Kurzinfo: Turtle Graphics

Programmart: Basic-Erweiterung mit neuen Grafikbefehlen
Laden: LOAD "TURTLE GRAPHICS",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Freihändiges Zeichnen ist ebenso möglich wie programmierte Grafikanweisungen
Benötigte Blocks: 9
Programmautor: Peter Menke

Demoprogramm und Speicheraufteilung

Alle neuen Befehle lassen sich unabhängig vom Einschaltzustand des Grafikbildschirms anwenden.

Unser Programmbeispiel zeigt, wie die Anweisungen in einem Listing funktionieren. Laden Sie das Demo mit:

```
LOAD "TURTLE DEMO",8
```

und starten Sie mit RUN.

Achtung: Das Beispiel läuft nur, wenn Sie vorher »Turtle Graphics« geladen und aktiviert haben!

Es zeigt diverse Grafiken (Roboterkopf, Linien, Rechtecke und Spiralen, Abb. 2). Außerdem bietet das Demoprogramm die Möglichkeit, per Joystick oder Lightpen freihändig zu zeichnen. Beide Modi verläßt man mit <F7>. Zuletzt wird ein Muster von zufällig gesetzten Pixeln auf dem Bildschirm verteilt. Per <RUN/STOP> können Sie die Grafik-Show abbrechen. Ab sofort befinden Sie sich wieder im Editormodus des Textbildschirms. Sie können nun das alte Basic-Programm mit NEW löschen und ein neues eingeben.

Hier einige Tips für Programmierer: Das Video-RAM des Textbildschirms liegt bei $0400 (1024). Der Hires-Bereich beginnt bei $CC00, das Farb-RAM für die Grafik wurde ab Adresse $C800 eingerichtet. »Turtle Graphics« selbst liegt im Bereich von $C000 bis $C88B. Die neue Belegung der Funktionstasten wird durch Anzapfen des Interrupts verwirklicht. Die Textfenster erzeugt man durch Manipulation des Rasterzeilen-Interrupts.

Der Programmeinsprung liegt bei $0820 (2080). Das wird durch die Basic-Zeile zu Beginn des Maschinenprogramms veranlaßt. Nach dem Start verschiebt eine Programmroutine im Bereich $0820 bis $083D die Daten in den Bereich ab $C000 (49152). Sie können »Turtle Graphics« jederzeit mit <RUN/STOP> verlassen, durch SYS 49152 läßt sich das Programm wieder aufrufen und starten.

Wir wünschen viel Spaß beim Programmieren des schnellen Grafikcursors! (Peter Menke/bl)



64'er Sonderhefte alle auf einen Blick

Die 64'er Sonderhefte bieten Ihnen umfassende Information in komprimierter Form zu speziellen Themen rund um die Commodore C 64 und C 128. Ausgaben, die eine Diskette enthalten, sind mit einem Diskettensymbol gekennzeichnet.

C 64, C 128, EINSTEIGER



SH 0022: C 128 III
Farbiges Scrolling im 80-Zeichen Modus / 8-Sekunden-Kopierprogramm



SH 0026: Rund um den C64
Der C64 verständlich für Alle mit ausführlichen Kursen



SH 0029: C 128
Starke Software für C 128/ C 128D / Alles über den neuen C 128D im Blechgehäuse





SH 0036: C 128
Power 128: Directory komfortabel organisieren / Haushaltsbuch: Finanzen im Griff / 3D-Landschaften auf dem Computer



SH 0038: Einsteiger
Alles für den leichten Einstieg / Super Malprogramm / Tolles Spiel zum selbermachen / Mehr Spaß am Lernen

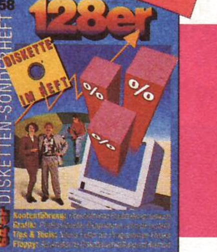




SH 0050: Starthilfe
Alles für den leichten Einstieg / Heiße Rythmen mit dem C 64 / Fantastisches Malprogramm



SH 0051: C 128
Volle Floppy-Power mit "Rubikon" / Aktienverwaltung mit "Börse 128"



SH 0058: 128er
Übersichtliche Buchhaltung zuhause / Professionelle Diagramme



SH 0062: Erste Schritte
RAM-Exos: Disketten superschnell geladen/ Exbasic Level II: über 70 neue Befehle/ Rafinessen mit der Tastatur



SH 0064: 128ER
Anwendungen: USA Journal / Grundlagen: CP/M, das dritte Betriebssystem/ VDC-Grafik: Vorhang auf für hohe Auflösung

GEOS, DATEIVERWALTUNG



SH 0028: Geos / Dateiverwaltung
Viele Kurse zu Geos / Tolle Geos-Programme zum Abtippen



SH 0048: GEOS
Mehr Speicherplatz auf Geos-Disketten / Schneller Texteditor für Geowrite / Komplettes Demo auf Diskette



SH 0059: GEOS
GeoBasic: Großer Programmierkurs mit vielen Tips & Tricks

PROGRAMMIERSPRACHEN



SH 0035: Assembler
Abgeschlossene Kurse für Anfänger und Fortgeschrittene



SH 0040: Basic
Basic Schritt für Schritt / Keine Chance für Fehler / Profi-Tools und viele Tips

ANWENDUNGEN



SH 0031: DFÜ, Musik, Messen-Steuern-Regeln
Alles über DFÜ / BTX von A-Z / Grundlagen / Bauanleitungen



SH 0046: Anwendungen
Das erste Expertensystem für den C 64 / Bessere Noten in Chemie / Komfortable Dateiverwaltung



SH 0056: Anwendungen
Gewinnauswertung beim Systemlotto / Energieverbrauch voll im Griff / Höhere Mathematik und C64

TIPS, TRICKS & TOOLS



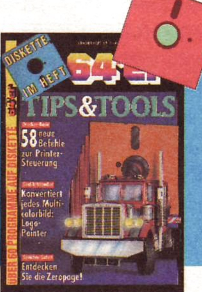
SH 0024: Tips, Tricks& Tools
Die besten Peeks und Pokes sowie Utilities mit Pfiff



SH 0043: Tips, Tricks&Tools
Rasterinterrupts - nicht nur für Profis / Checksummer V3 und MSE / Programmierhilfen



SH 0057: Tips & Tricks
Trickreiche Tools für den C64 / Drucker perfekt installiert



SH 0065: Tips&Tools
Streifzug dirch die Zeropage/ Drucker-Basic: 58 neue Befehle zur Printer-Steuerung/ Multicolorgrafiken konvertieren/ über 60 heiße Tips&Tricks

HARDWARE



SH 0025: Floppylaufwerke
Wertvolle Tips und Informationen für Einsteiger und Fortgeschrittene



SH 0032: Floppylaufwerke und Drucker
Tips&Tools / RAM-Erweiterung des C64 / Druckerroutinen



SH 0047: Drucker, Tools
Hardcopies ohne Geheimnisse / Farbige Grafiken auf s/w-Druckern



SH 0067: Wetterstation:
Temperatur, Luftdruck und feuchte messen/ DCF-Funkuhr und Echtzeituhr/ Daten konvertieren: vom C64 zum Amiga, Atari ST und PC

DTP



SH 0039: DTP, Textverarbeitung
Komplettes DTP-Paket zum Abtippen / Super Textsystem / Hochauflösendes Zeichenprogramm

GRAFIK



SH 0020: Grafik
Grafik-Programmierung / Bewegungen



SH 0045: Grafik
Listings mit Pfiff / Alles über Grafik-Programmierung / Erweiterungen für Amica-Paint



SH 0055: Grafik
Amica-Paint: Malen wie ein Profi / DTP-Seiten vom C64 / Tricks&Utilities zur Hires-Grafik

SH 0063: Grafik
Text und Grafik mischen ohne Flimmern / EGA: Zeichenprogramm der Superlative / 3 professionelle Editoren



SH 0068: Anwendungen
Kreuzworträtsel selbstgemacht/ Happy Synth: Super-Synthesizer für Sound-Freaks/ Der C64 wird zum Planetarium/ Sir-Compact: Bit-Packer verdichtet Basic- und Assemblerprogramme

SPIELE



SH 0030: Spiele für C 64 und C 128
Spiele zum Abtippen für C 64/ C 128 / Spieleprogrammierung



SH 0037: Spiele
Adventure, Action, Geschicklichkeit / Profihilfen für Spiele / Überblick, Tips zum Spielekauf



SH 0042: Spiele
Profispiele selbst gemacht / Adventure, Action, Strategie



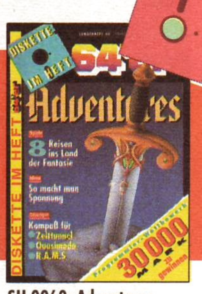
SH 0049: Spiele
Action, Adventure, Strategie / Sprites selbst erstellen / Virenkiller gegen verseuchte Disketten



SH 0052: Abenteuerspiele
Selbstprogrammieren: Von der Idee zum fertigen Spiel / So knacken Sie Adventures



SH 0054: Spiele
15 tolle Spiele auf Diskette/ der Sieger unseres Programmierwettbewerbs: Crillion II/ ein Cracker packt aus: ewige Leben bei kommerziellen Spielen



SH 0060: Adventures
8 Reisen ins Land der Fantasie - so macht Spannung Spaß



Top Spiele 1
Die 111 besten Spiele im Test/ Tips, Tricks und Kniffe zu heißen Games/ Komplettlösung zu "Last Ninja II"/ große Marktübersicht: die aktuellen Superspiele für den C64



SH 0061: Spiele
20 heiße Super Games für Joystick-Akrobaten/ Cheat-Modi und Trainer POKEs zu über 20 Profi-Spielen/ Krieg der Kerne: Grundlagen zur Spielerprogrammierung



SH 0066: Spiele
15 Top-Spiele mit Action und Strategie/ Mondlandung: verblüffend echte Simulation und Super-Grafik/ High-Score-Knacker: Tips&Tricks zu Action-Games

64'er Magazin auf einen Blick

Diese 64'er-Ausgaben bekommen Sie noch bei Markt&Technik für jeweils 7,- DM. Die Preise für Sonderhefte und Sammelbox entnehmen Sie bitte dem Bestellcoupon. Tragen Sie Ihre Bestellung im Coupon ein und schicken Sie ihn am besten gleich los, oder rufen Sie einfach unter 089 - 20 25 15 28 an.

11/90: Bausatztest: Der Taschengeldplotter / Vergleichstest: Drucker der Spitzenklasse / 5 Schnellbauschaltungen

12/90: Abenteuer BTX / Multitasking für C64 / Großer Spieleschwerpunkt / Programmierwettbewerb: 30 000 DM zu gewinnen

1/91: Die Besten Tips&Tricks / Neu: Reparaturecke / Floppy-Flop: Betriebssystem überlistet / Jahresinhaltsverzeichnis

2/91: Sensation: Festplatte für den C 64 / Drucken ohne Ärger / Listing des Monats: Actionspiel "Ignition" / Longplay: Dragon Wars

3/91: Bauanleitung: Universelles Track-Display / Alles über Module für den C 64 / Festplatte HD 20 unter GEOS

4/91: Spiele-Schwerpunkt: 100 Tips, News, Tests / Neu: Grafikkurs / Fischer-Baukästen / Bauanleitung: Digitizer

5/91: Ätzanlage unter 50.- DM / GRB-Monitor am C64 / Longplay: Bard's Tale / Reparaturkurs: Die neuen C64 / Piratenknacker

6/91: C64er-Meßlabor: Universell erweiterungsfähig / Test: Pocket-Wrighter 3.0 - Bestes C64 Textprogramm / Listing des Monats: Autokosten im Griff

7/91: Trickfilm mit dem C 64 / Bauanleitung: 1541-Floppy mit Batteriebetrieb / Listing des Monats: Basic-Butler

8/91: Drucker unter 1000 DM / Test: GEO-RAM / Listing des Monats: 80-Farben-Malprogramm / Longplay: Secret of the Silver Plate

09/91: Joystick im Test / Die üblen Tricks mit Raubkopien / Die besten Drucker unter 1500 DM / Mit großem Spieleteil

10/91: 100 besten Tips&Tricks / Listing: Fraktal-Programm / C-64-Meßlabor: komfortables Kontrollmodul

11/91: Alles über Diskette & Floppy / Bauanleitung: C 64 steuert Laserstrahl / Sha-Jongg: Topspiel mit Spitzengrafik / Großer Spieleteil

12/91: Alle Spiele 1991 / Tolle Tips&Tricks für den C64 und C128 / Geschenktips für Computerfans / Komfortable Videoverwaltung

BESTELLCOUPON

Ich bestelle ____ 64er Sonderhefte Nr. ______________ ____ **DM**
zum Preis von je: 14,- DM (Heft ohne Diskette) ____ **DM**
16,- DM (Heft mit Diskette) ____ **DM**
9,80 DM (SH "Top Spiele 1") ____ **DM**
24,- DM (für die Sonderhefte 0051/ 0058/ 0064) ____ **DM**

Ich bestelle ____ 64er Magazin Nr. ____/____/____ ____ **DM**
zum Preis von je 7,- DM

Ich bestelle ____ Sammelbox(en) ____ **DM**
zum Preis von je 14,- DM

Gesamtbetrag ____ **DM**

Ich bezahle den Gesamtbetrag zzgl. Versandkosten nach Erhalt der Rechnung.

Name, Vorname ______________

Straße, Hausnummer ______________

PLZ, Wohnort ______________

Telefon (Vorwahl) ______________ Ich erlaube Ihnen hiermit mir interessante Zeitschriftenangebote auch telefonisch zu unterbreiten (ggf. streichen).

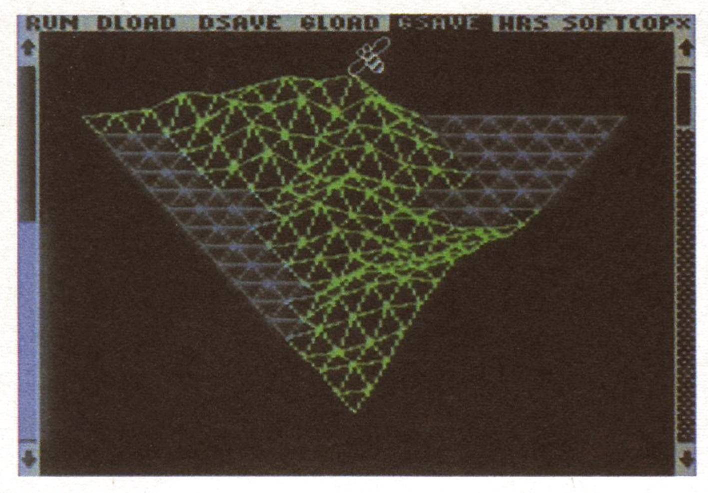
Schicken Sie bitte den ausgefüllten Bestellcoupon an: 64er Leserservice, CSJ, Postfach 140 220, 8000 München 5, Telefon 089/ 20 25 15 28



Superfrac 64 - Grand Canyon

Mathematik mit Romantik: »Superfrac 64« holt schroffe Gebirge und stille Seen aus den Tiefen des C64 an die Bildschirmoberfläche.

Äußerst beliebt bei Grafik-Freaks: Fraktale (sinngemäß übersetzt: Ausschnitte der Erdoberfläche), geschaffen aus exakten Koordinaten und mathematischen Berechnungsformeln, die das Programm bereits enthält. Lediglich für die Koordinateneingabe sind Sie zuständig. In wenigen Sekunden lassen sich nach korrekten Werten faszinierende Landschaften erzeugen, die der Natur nachempfunden sind.



[1] Alle Menüpunkte lassen sich mit dem Joystick Port 2 aktivieren

Laden Sie das Grafikprogramm mit:

```
LOAD "SUPERFRAC 64",8
```

und starten Sie mit RUN.

Nach dem Laden des Maschinensprache-Teils bietet das Programm eine komfortable Benutzeroberfläche (Abb. 1), für die Sie den Joystick in Port 2 brauchen. Der Grundriß einer Demofraktalgrafik ist bereits vorgegeben (aus dem Disketten-File »DAT.SF 64«).

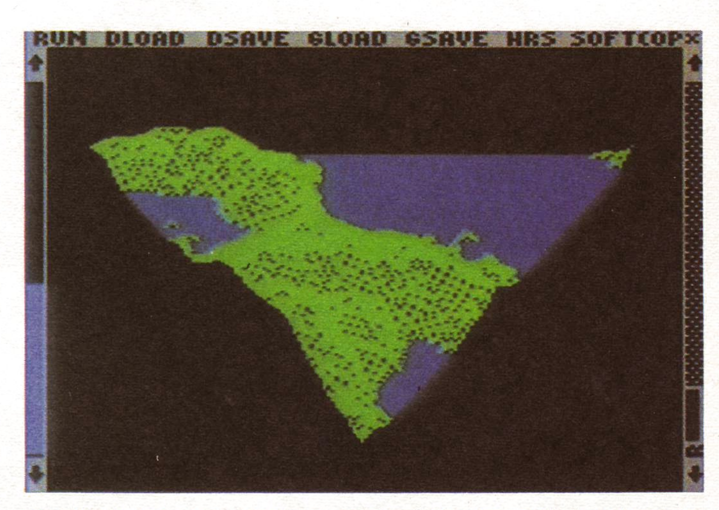
Die Funktionen stehen in der oberen Menüleiste, die man mit dem Mauszeiger und Druck auf den Feuerknopf aktiviert:

RUN

Für ein neues Fraktalbild müssen Sie jetzt per Tastatur die Höhen der sechs Stützpunkte eingeben, die anschließende Skizze zeigt ihre Verteilung (Abb. 2). Der somit neu entstandene Grundriß erscheint auf dem Bildschirm. Jetzt treten die beiden seitlichen Bildleisten in Aktion, die man per Knopfdruck nach oben oder unten verschieben kann:

In der rechten Leiste befindet sich ein schwarzer Schieberegler, der per Mausklick Schritt für Schritt in Richtung des unteren Pfeils verschoben werden muß. Die Bildschirmgrafik paßt sich an. Ist der Regler ganz unten, bringt der Bildschirm das Fraktal in der maximalen Auflösung.

Die linke Leiste ist in der oberen Hälfte schwarz, in der unteren blau. Damit stellt man den Blickwinkel ein, aus dem sich die Computerlandschaft betrachten läßt: Je höher der Blauanteil, desto tiefer der Blickwinkel.



[2] Wie hoch das Computergebirge werden soll, bestimmen Sie mit den Eckpunktwerten

Kurzinfo: Superfrac 64

Programmart: Fraktalgrafik
Laden: LOAD "SUPERFRAC 64",8
Starten: nach dem Laden mit RUN
Besonderheiten: Sichtwinkel und Grafikdichte individuell einstellbar
Benötigte Blocks: 29
Programmautor: Ingmar Camphausen

DLOAD

lädt Grafiken, deren Koordinaten-Daten auf Diskette gespeichert wurden (nicht der Inhalt des Hires-Bildschirms!). Diese Dateien beginnen mit einem reversen »D«.

DSAVE

speichert die Koordinatendaten auf Diskette.

GLOAD

lädt ein Grafikbild von Diskette, das als Hires-Bild gespeichert wurde (33 Blöcke).

GSAVE

speichert ein neues Fraktal als hochauflösende Grafik auf Diskette (33 Blöcke).

HRS/MUL

gibt Auskunft über den aktivierten Grafikmodus: HRS = Hires (einfarbig), MUL = Multicolor (mehrfarbig). Der letztgenannte Modus ist bei Programmstart voreingestellt.

SOFTCOP

Damit stellt man den Softcopy-Modus auf dem Bildschirm ein: Befindet sich hinter der Anzeige ein Haken, wird der Bildschirm mit der vorherigen Fraktaleinteilung nicht gelöscht: Die neuberechnete Grafik wird sanft drüberkopiert. Steht hinter dem Menüpunkt ein Kreuz, wird der Bildschirm vor jedem neuen Grafikaufbau gelöscht.

Lassen Sie sich überraschen, welche wildromantischen Landschaften in Ihrem Computer schlummern!

(Ingmar Camphausen/bl)

Charmaster - komfortabler Zeichensatzeditor

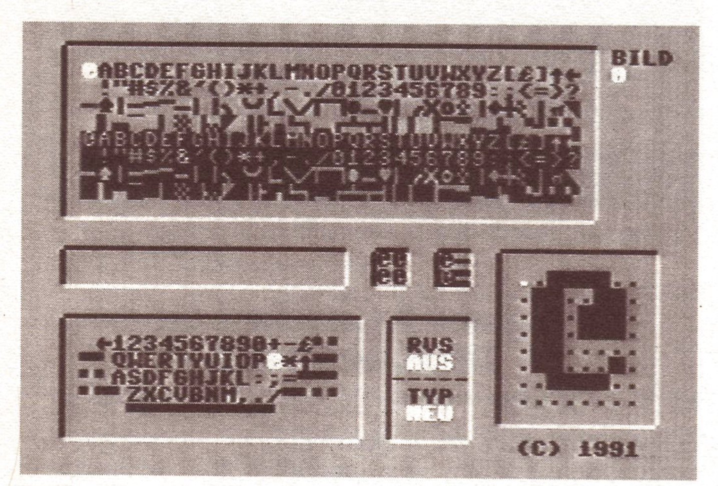
VERSCHNÖRKELT ODER COOL

Schöne Zeichensätze kaufen, kann jeder. Selbermachen ist angesagt. Damit das aber nicht zur Quälerei wird, ein Tip: Charmaster!

Wie Sie einen Zeichensatz über PEEKs und POKEs konstruieren, lesen Sie ab Seite 28 nach. Es geht aber auch einfacher. Mit einem Hilfsprogramm, das dafür eine Benutzeroberfläche besitzt. Mit ihm lassen sich alle 256 Bildschirmzeichen auf einen Blick bearbeiten, drehen, invertieren usw. Geladen wird von der beiliegenden Diskette mit

```
LOAD"CHARMASTER",8
```

und gestartet mit RUN. Zuerst wird der Basicspeicher verschoben, danach das Hauptprogramm »CM MAIN« geladen. Dieses wiederum lädt den Assembler-Teil nach. Er beschleunigt später die zeitkritischen Routinen. Nun werden die Tabelle für die Tastaturanzeige und der Bildschirm aufgebaut. Währenddessen flimmert der Bildschirm in allen Farben. Nach etwa vier Sekunden zeigt sich die Benutzeroberfläche (Abb. 1).



[1] Die übersichtliche Benutzeroberfläche erlaubt vielseitige Bearbeitung der Zeichensätze

Die Benutzeroberfläche

Der Bildschirmcode wird zunächst dezimal ausgegeben (unter »Bild«), zeigt aber nach <C> auch den Wert in hexadezimaler Form an. Zugleich wird der ASCII-Code dezimal/hexadezimal dargestellt. Da die Umrechnung in diese Form eine gewisse Zeit benötigt, ist diese Anzeige abschaltbar.

Im Tastaturfeld (unten links) wird ständig die Tastenkombination auf der Tastatur gezeigt.

Das linke der beiden kleinen Felder in der Mitte dient zur Kontrolle beim Entwurf von Musterzeichen.

Das Feld daneben zeigt die zum Zeichen gehörende Vierergruppe bei 2x2-Zeichen.

Zusätzlich ist angegeben, ob das Zeichen zu den reversen oder normalen Zeichen gehört (RVS EIN/AUS).

Die Angabe darunter zeigt den Ausgabetyp. Er wird mit <T> verändert.

Das größte Feld (oben) verschafft einen Überblick über alle Zeichen. Natürlich ist auch hier das gewählte Zeichen farblich hervorgehoben.

Im Editierfeld (unten rechts) erscheint das gewählte Zeichen in vergrößerter Form. In diesem Feld wird es auch beliebig modifiziert. Die Zeichen für gesetzten oder gelöschten Punkt lassen sich beliebig ändern (<B>).

Das längliche Feld in der Mitte dient zur Ausgabe von Funktionsbezeichnungen, Sicherheitsabfragen usw.. Diese Meldungen erscheinen jeweils für ein paar Sekunden, außer wenn auf Eingaben gewartet wird.

Bewegen in den Editierfeldern

Zwei unterschiedliche Cursor stehen zur Verfügung. Der Editiercursor zeigt die Position im Editierfeld, der Zeichencursor das gerade im Editierfeld abgebildete Zeichen:

<**Cursortasten**> - bewegen den Editiercursor

<**.**>, <**SHIFT .**>, <**/**> und <**SHIFT /**> - stehen für den Zeichencursor (Feld oben) zur Verfügung.

Zur direkten Anwahl der Zeichen existiert ein Suchmodus:

<**S**> - Suchmodus für nichtreverse Zeichen

... danach wartet Charmaster auf eine weitere Taste. Drücken Sie die entsprechende Kombination für das anzuspringende Zeichen.

<**SHIFT S**> - Suchmodus für reverse Zeichen

... danach wartet Charmaster wieder auf die Tastenkombination.

Manipulation eines Zeichens

Jedes (rechts unten) angezeigte Zeichen kann beliebig bearbeitet werden. Für einzelne Punkte dienen:

<**RETURN**> - Punkt setzen, löschen oder wechseln

... was geschieht ist abhängig vom Modus

<**P**> - Modus wechseln

... bei jedem Tastendruck wird der jetzt gültige Modus im Infofeld gezeigt.

Zusätzlich lassen sich die kompletten Zeichen per Tastendruck manipulieren:

<**L**> - löschen
<**F**> - füllen
<**I**> - invertieren
<**M**> - schraffieren
<**SHIFT M**> - Zeichenhintergrund schraffieren
<**Q**> - drehen im Uhrzeigersinn
<**SHIFT Q**> - drehen entgegen dem Uhrzeigersinn
<**SHIFT 4**> - nach oben herausscrollen
<**SHIFT D**> - nach unten herausscrollen
<**SHIFT E**> - nach links herausscrollen
<**SHIFT R**> - nach rechts herausscrollen
<**4**> - nach oben scrollen
<**D**> - nach unten scrollen
<**E**> - nach links scrollen
<**R**> - nach rechts scrollen
<**X**> - an der X-Achse spiegeln
<**Y**> - an der Y-Achse spiegeln

Bei den reinen Scrollfunktionen gehen keine Informationen verloren, wogegen beim Herausscrollen das Zeichen in Scrollrichtung verschwindet.

Verändern von Zeichenfolgen

Charmaster erlaubt auch die Veränderung von Zeichenfolgen. Beispielsweise lassen sich alle Zeichen untereinander vertauschen.

Zeichen tauschen

Dazu definiert man zuerst eines der Tauschzeichen mit <SHIFT V>, bewegt den Zeichencursor auf das zu tauschende Zeichen und drückt <V>.

Taste	UNDO	Funktion
<A>	1	Zwischenspeicher mit Zeichen AND verknüpfen
<B>		Edit-Feld-Zeichen verändern
<SHIFT B>		CHARMASTER verlassen
<C>		Alle Codes anzeigen (ein/aus)
<D>	1	Zeichen nach unten durchscrollen
<SHIFT D>	1	Zeichen nach unten herausscrollen
<E>	1	Zeichen nach links durchscrollen
<SHIFT E>	1	Zeichen nach links herausscrollen
<F>	1	Zeichen ausfüllen
<G>		Bereich dem entsprechenden reversen/nicht reversen angleichen
<H>	1	Originalzeichen wiederherstellen
<I>		12 Zeichen invertieren
<J>		Strichcursorfunktion
<K>		Zwischenspeicher kopieren
<L>	1	Zeichen löschen
<M>	1	Zeichen schraffieren
<SHIFT M>	1	Zeichenhintergrund schraffieren
<N>		Normalzeichensatz wiederherstellen
<O>	1	Zwischenspeicher mit Zeichen OR verknüpfen
<P>		Umschalten zwischen Punkt setzen/löschen/wechseln
<Q>	1	Zeichen im Uhrzeigersinn drehen
<SHIFT Q>	1	Zeichen gegen den Uhrzeigersinn drehen
<R>	1	Zeichen nach rechts durchscrollen
<SHIFT R>	1	Zeichen nach rechts herausscrollen
<S>		Suchmodus nicht revers
<SHIFT S>		Suchmodus revers
<T>		Ausgabetyp variieren (groß/klein/editierbar)
<U>		Undofunktion ein/aus
<V>	2	Zeichen tauschen
<SHIFT V>		1.Tauschzeichen festlegen
<W>		Tastenwiederholungsautomatik ein/aus
<X>		12 Zeichen an der X-Achse spiegeln
<Y>		12 Zeichen an der Y-Achse spiegeln
<Z>		Zeichen in den Zwischenspeicher kopieren
<4>	1	Zeichen nach oben durchscrollen
<SHIFT 4>	1	Zeichen nach oben herausscrollen
<RETURN>	1	Punkt nach »P« bearbeiten
<HOME>		Zeichencursor in Homeposition bringen
<DEL>		Edit-Cursor in Homeposition bringen
<.>/?>		Zeichencursor bewegen
<CRSR>		Edit-Cursor bewegen
<F1>		Lademodus
<F3>		Speichermodus
<F5>		Diskbefehl senden
<F7>	1/2	UNDO

Kurzbeschreibung der Befehle: In der zweiten Spalte bezeichnet die Zahl die Art der UNDO-Funktion. 1=UNDO-fähig, 2= durch sich selbst UNDO-fähig.

Zeichensatz wechseln

Falls der Zeichensatz nicht mehr gefällt, kann einer der Normalzeichensätze gewählt werden. <H> stellt die Originalzeichen wieder her.

Mühselige Kleinarbeit wird durch die Blockfunktion »angleichen« erspart: Ein Zeichensatz im nichtreversen Teil entworfen läßt sich mühelos invertieren oder andersherum.

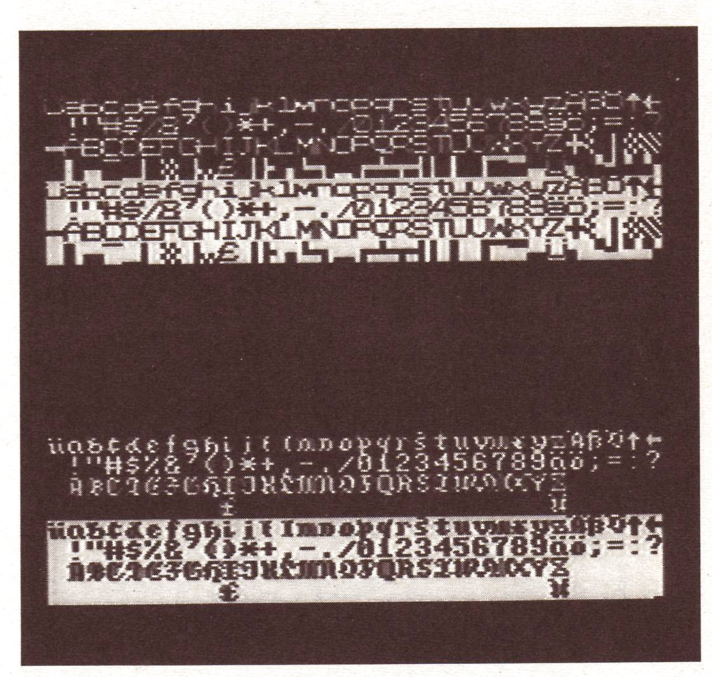
<G> - Invertieren einer Zeichefolge

... zuerst wird der Bildschirmcode des ersten anzugleichenden Zeichens und dann nach der des letzten abgefragt. Angaben, die keinen Sinn ergeben (z.B. 120-140, da sowohl im nichtreversen als auch im reversen Teil) werden ignoriert.

Zur professionellen Gestaltung dient die Strichcursor-Funktion.

<J> - Strichcursor für Zeichensätze.

Weiterhin besteht die Möglichkeit die Tastenwiederholungsautomatik ein- bzw. auszuschalten. Wenn z.B. der Punkt oder das reverse SPACE verändert werden sollen, besteht auch die Möglichkeit, mit <B> die Edit-Feld-Zeichen zu verändern.



[2] Zwei der geänderten Zeichensätze auf Diskette

Logische Verknüpfung im Zwischenspeicher

Für Verknüpfungen mit anderen Zeichen existiert ein Zwischenspeicher. In ihm kann jeweils ein Zeichen bearbeitet werden. Mit den erlaubten Verknüpfungsfunktionen lassen sich besonders ansprechende Effekte erzielen. Beispiel: Sie entwerfen ein Strichmuster und nehmen dieses durch AND aus dem Zeichen heraus oder fügen es durch OR dem Zeichen hinzu:

<Z> - kopiert ein Zeichen in den Zwischenspeicher

<K> - kopiert den Zwischenspeicher in das gewählte Zeichen

<O> - verknüpft den Inhalt des Zwischenspeichers OR mit dem gewählten Zeichen.

<A> - verknüpft den Inhalt des Zwischenspeichers AND mit dem gewählten Zeichen.

Beachten Sie, daß der Zwischenspeicher jeweils nur ein Zeichen enthalten kann. Die Verknüpfung bei diesen Befehlen geschieht immer im Zeichen, über dem der Cursor steht. Dabei bleibt der Inhalt (außer bei <Z>) unberührt.

Die UNDO-Funktion

Die meisten Funktionen sind UNDO-fähig, außer solchen, die sich schon durch Wiederholung zurücknehmen lassen. Beispiel: Bestimmen Sie zuerst eines der Tauschzeichen (<SHIFT V>). Dann bewegen Sie den Zeichencursor auf das zu tauschende Zeichen und drücken <V>. Die Zeichen werden vertauscht. Ein nochmaliges <V> bringt die Zeichen wieder an ihren ursprünglichen Ort.

Ansonsten nimmt <**F7**> den letzten Schritt zurück. UNDO wird über <**U**> ein- bzw. ausgeschaltet.

Disketten-Operationen

<**F1**> - Laden eines Zeichensatzes

Die Ladefunktion benötigt PRG-Files. Bei einigen Floppys wird das File nicht ordnungsgemäß geschlossen. Warten Sie in so einem Fall ca. eine Minute, dann unterbrechen Sie mit <RUN/STOP>. Danach geben Sie ein:

```
CLOSE1: GOTO 6028
```

<**F3**> - Speichern eines Datensatzes

Gespeichert wird nach Angabe der Startadresse. Die vorgegebene Speicheradresse (2048) ermöglicht den Zeichensatz außerhalb des Programms zu laden und mit »POKE 53272,19« aufzurufen.

<**F5**> - Diskettenkommando

... sendet einen Diskbefehl direkt ans Laufwerk. »$« als Eingabe zeigt den Disketteninhalt

<**SHIFT B**> beendet das Programm nach einer Sicherheitsabfrage.

Achtung: Das Hauptprogramm von Charmaster ist in Basic geschrieben und läßt sich mit <RUN/STOP> unterbrechen. Diese Vorgehensweise ist allerdings nicht empfehlenswert, da beim Neustart mit RUN alle Variablenpointer gelöscht sind, ähnlich dem neuen Laden des Programms.

Wenn Sie »Charmaster« auf Ihre Arbeitsdiskette kopieren, benötigen Sie drei Programme von der beiliegenden Diskette:

1. »CHARMASTER« - Ladeprogramm zur Verschiebung des Basicspeichers
2. »CM MAIN« - Basic-Hauptprogramm
3. »CM $C000« - Maschinenspracheprogramm

Zusätzlich stehen Ihnen 14 geänderte Zeichensätze zur Verfügung (Abb. 2), mit denen Sie experimentieren können: Die Bezeichnungen auf Diskette lauten: »20«, »23«, »24«, »25«, »26«, »29«, »30«, »37«, »38«, »39«, »40«, »48«, »49« und »50«. Alle Sätze sind neun Blocks lang und lassen sich vom Charmaster aus bearbeiten. Beachten Sie hierzu auch die Hinweise unter <F1>. (gr)

Kurzinfo: Charmaster

Programmart: Zeichen-Editor
Laden: LOAD"CHARMASTER",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: lädt »CM MAIN« und »CM$C000« nach
Benötigte Blocks: 56
Programmautor: Ingo Dellwig

Kurzinfo: Zeichensätze

Programmart: geänderte C64 Zeichensätze
Laden: im Charmaster
Benötigte Blocks: je 9
Programmautor: Stefan Assauer

So finden Sie die Programme auf der Diskette

DISKETTE SEITE 1

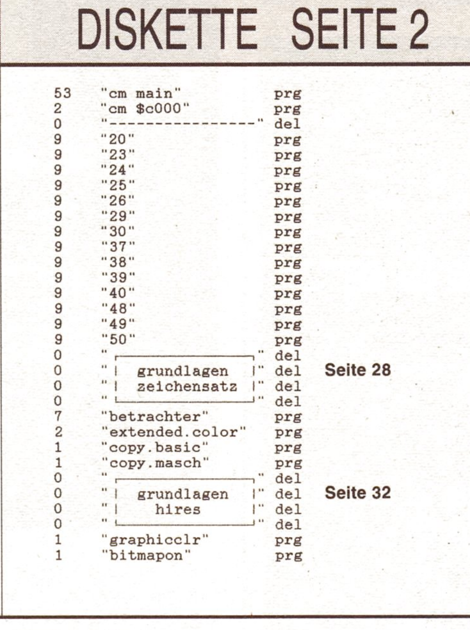
```
0  "64'er sonderh.75" s1 2a
27    "disklader"          prg   Seite 23
0     "┌──────────────┐"    del
0     "│ zeichenprogr.│"    del
0     "└──────────────┘"    del
5     "hi-eddi"            prg   Seite 40
19    "hi-exe"             prg
2     "hi-print"           prg
0     "----------------"   del
37    "menue"              prg
33    "demo1.pic"          prg
33    "demo2.pic"          prg
33    "mot1.pic"           prg
33    "mot2.pic"           prg
33    "mot3.pic"           prg
33    "mot4.pic"           prg
33    "mot5.pic"           prg
33    "mot6.pic"           prg
0     "----------------"   del
4     "grafiklader"        prg
```

```
19    "moviecreator"       prg
8     "grapher"            prg
2     "bmc"                prg
0     "┌──────────────┐"    del
0     "│ erweiterung  │"    del
0     "└──────────────┘"    del
16    "blitter"            prg   Seite 7
5     "blitter demo1"      prg
2     "blitter demo2"      prg
0     "----------------"   del
9     "turtle graphics"    prg   Seite 10
9     "turtle demo"        prg
0     "----------------"   del
17    "interlace 64"       prg   Seite 4
2     "farb.demo"          prg
51    "split.demo"         prg
```

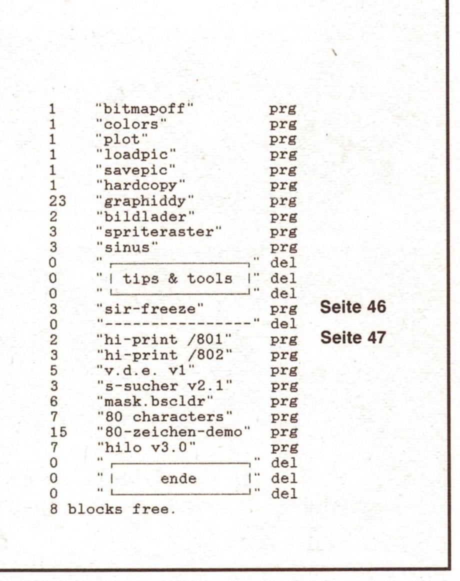
```
0     "┌──────────────┐"    del
0     "│   sprites    │"    del
0     "└──────────────┘"    del
3     "sprite-moover"      prg   Seite 26
4     "demo.moover"        prg
0     "----------------"   del
8     "sprites"            prg
0     "┌──────────────┐"    del
0     "│ intro-demos  │"    del   Seite 36
0     "└──────────────┘"    del
44    "flexgrid"           prg
51    "plexer dycp"        prg
7     "bob plotting"       prg
46    "1-pixel fld"        prg
0     "┌──────────────┐"    del
0     "│   diskette   │"    del
0     "│  beidseitig  │"    del
0     "│   bespielt   │"    del
0     "└──────────────┘"    del
3 blocks free.
```

DISKETTE SEITE 2

```
0  "64'er sonderh.75" s2 2a
27    "disklader"          prg   Seite 23
0     "┌──────────────┐"    del
0     "│   sprites    │"    del
0     "└──────────────┘"    del
14    "spritemon 2.2"      prg   Seite 24
0     "┌──────────────┐"    del
0     "│   fraktale   │"    del
0     "└──────────────┘"    del
8     "superfrac 64"       prg   Seite 14
18    "mc.sf 64"           prg
3     "dat.sf 64"          prg
18    "■bucht"             prg
18    "■inseln"            prg
18    "■felsmassiv"        prg
18    "■landbruecken"      prg
33    "■bucht.35 m < nn"   prg
33    "■bucht.25 m < nn"   prg
33    "■bucht.10 m < nn"   prg
33    "■bucht 0 m nn"      prg
33    "■bucht.10 m > nn"   prg
33    "■kanalinsel"        prg
33    "■bergruecken"       prg
0     "┌──────────────┐"    del
0     "│ zeichensatz  │"    del
0     "└──────────────┘"    del
1     "charmaster"         prg   Seite 15
```



```
53    "cm main"            prg
2     "cm $c000"           prg
0     "----------------"   del
9     "20"                 prg
9     "23"                 prg
9     "24"                 prg
9     "25"                 prg
9     "26"                 prg
9     "29"                 prg
9     "30"                 prg
9     "37"                 prg
9     "38"                 prg
9     "39"                 prg
9     "40"                 prg
9     "48"                 prg
9     "49"                 prg
9     "50"                 prg
0     "┌──────────────┐"    del
0     "│  grundlagen  │"    del   Seite 28
0     "│ zeichensatz  │"    del
0     "└──────────────┘"    del
7     "betrachter"         prg
2     "extended.color"     prg
1     "copy.basic"         prg
1     "copy.masch"         prg
0     "┌──────────────┐"    del
0     "│  grundlagen  │"    del   Seite 32
0     "│    hires     │"    del
0     "└──────────────┘"    del
1     "graphicclr"         prg
1     "bitmapon"           prg
```



```
1     "bitmapoff"          prg
1     "colors"             prg
1     "plot"               prg
1     "loadpic"            prg
1     "savepic"            prg
1     "hardcopy"           prg
23    "graphiddy"          prg
2     "bildlader"          prg
3     "spriteraster"       prg
3     "sinus"              prg
0     "┌──────────────┐"    del
0     "│ tips & tools │"    del
0     "└──────────────┘"    del
3     "sir-freeze"         prg   Seite 46
0     "----------------"   del
2     "hi-print /801"      prg   Seite 47
3     "hi-print /802"      prg
5     "v.d.e. v1"          prg
3     "s-sucher v2.1"      prg
6     "mask.bscldr"        prg
7     "80 characters"      prg
15    "80-zeichen-demo"    prg
7     "hilo v3.0"          prg
0     "┌──────────────┐"    del
0     "│     ende     │"    del
0     "└──────────────┘"    del
8 blocks free.
```

WICHTIGE HINWEISE

zur beiliegenden Diskette:

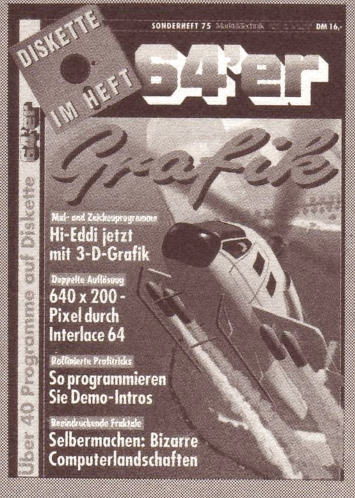
Aus den Erfahrungen der bisherigen Sonderhefte mit Diskette wollen wir ein paar Tips an Sie weitergeben:

1. **Bevor Sie mit den Programmen auf der Diskette arbeiten, sollten Sie unbedingt eine Sicherheitskopie der Diskette anlegen. Verwenden Sie dazu ein beliebiges Kopierprogramm, das eine komplette Diskettenseite dupliziert.**

2. **Auf der Originaldiskette ist wegen der umfangreichen Programme nur wenig Speicherplatz frei. Dies führt bei den Anwendungen, die Daten auf die Diskette speichern, zu Speicherplatz-Problemen. Kopieren Sie daher das Programm, mit dem Sie arbeiten wollen, mit einem File-Copy-Programm auf eine leere, formatierte Diskette und nutzen Sie diese als Arbeitsdiskette.**

3. **Die Rückseite der Originaldiskette ist schreibgeschützt. Wenn Sie auf dieser Seite speichern wollen, müssen Sie vorher mit einem Diskettenlocher eine Kerbe an der linken oberen Seite der Diskette anbringen, um den Schreibschutz zu entfernen. Probleme lassen sich von vornherein vermeiden, wenn Sie die Hinweise unter Punkt 2 beachten.**

ALLE PROGRAMME
aus diesem Heft



HIER

Herausgeber: Carl-Franz von Quadt, Otmar Weber
Redaktionsdirektor: Dr. Manfred Gindle

Chefredakteur: Georg Klinge (gk) – verantwortlich für den redaktionellen Teil
Stellv. Chefredakteur: Arnd Wängler (aw)
Textchef: Jens Maasberg
Redaktion: Harald Beiler (bl), Herbert Großer (gr)
Produktion: Andrea Pfliegensdörfer
Redaktionsassistenz: Sylvia Wilhelm, Birgit Misera

So erreichen Sie die Redaktion:
Tel. 089/4613-202, Telefax: 089/4613-5001, Btx: 64064

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, so muß das angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in den von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträgern. Mit Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß die Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Layout: Dagmar Portugall
Bildredaktion: Wallo Linne (Ltg.), Sabine Lechner, Roland Müller, Tina Steiner (Fotografie), Ewald Standke, Norbert Raab (Spritzgrafik), Werner Nienstedt (Computergrafik)

Anzeigendirektion: Jens Berendsen
Anzeigenleitung: Philipp Schiede
Anzeigenverwaltung und Disposition: Christopher Mark (421)

So erreichen Sie die Anzeigenabteilung:
Tel. 089/4613-494, Telefax: 089/4613-789

Gesamtvertriebsleiter: York von Heimburg
Vertriebsmarketing: Rainer Drumm

Vertrieb Handel: MZV, Moderner Zeitschriften Vertrieb GmbH & Co. KG, Breslauer Straße 5, Postfach 11 23, 8057 Eching, Tel. 089/3190 0613

Verkaufspreis: Das Einzelheft kostet DM 16,-

Produktion: Klaus Buck (Ltg./180), Wolfgang Meyer (Stellv./887)

Druck: SOV. Graphische Betriebe, Laubanger 23, 8600 Bamberg

Sonderdruck-Dienst: Alle in dieser Ausgabe erschienenen Beiträge sind in Form von Sonderdrucken erhältlich. Anfragen an Reinhard Jarczok, Tel. 089/4613-185, Telefax 089/4613-774

Vorstand: Otmar Weber (Vors.), Bernd Balzer, Dr. Rainer Doll, Lutz Glandt

Verlagsleitung: Wolfram Höfler
Operation Manager: Michael Koeppe

Direktor Zeitschriften: Michael M. Pauly

Anschrift des Verlags: Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052, Telefax 089/4613-100

ISSN 0931-8933

Mitteilung gem. Bayerischem Pressegesetz: Aktionäre, die mehr als 25% des Kapitals halten: Otmar Weber, Ingenieur, München; Carl-Franz von Quadt, Betriebswirt, München; Aufsichtsrat: Carl-Franz von Quadt (Vorsitzender), Dr. Robert Dissmann (stellv. Vorsitzender), Dr. Erich Schmitt.

Copyright-Erklärung

Name: ....

Anschrift: ....

Datum: ....

Computertyp: ....

Benötigte Erweiterung/Peripherie: ....

Datenträger: Kassette/Diskette ....

Programmart: ....

Ich habe das 18. Lebensjahr bereits vollendet

...., den ....

....
(Unterschrift)

Wir geben diese Erklärung für unser minderjähriges Kind als dessen gesetzliche Vertreter ab.

...., den ....

....

Bankverbindung:

Bank/Postgiroamt: ....

Bankleitzahl: ....

Konto-Nummer: ....

Inhaber des Kontos: ....

Das Programm/die Bauanleitung: ....

das/die ich der Redaktion der Zeitschrift 64'er übersandt habe, habe ich selbst erarbeitet und nicht, auch nicht teilweise, anderen Veröffentlichungen entnommen. Das Programm/die Bauanleitung ist daher frei von Rechten anderer und liegt zur Zeit keinem anderen Verlag zur Veröffentlichung vor. Ich bin damit einverstanden, daß die Markt & Technik Verlag AG das Programm/die Bauanleitung in ihren Zeitschriften oder ihren herausgegebenen Büchern abdruckt und das Programm/die Bauanleitung vervielfältigt, wie beispielsweise durch Herstellung von Disketten, auf denen das Programm gespeichert ist, oder daß sie Geräte und Bauelemente nach der Bauanleitung herstellen läßt und vertreibt bzw. durch Dritte vertreiben läßt.

Ich erhalte, wenn die Markt & Technik Verlag AG das Programm/die Bauanleitung druckt oder sonst verwertet, ein Pauschalhonorar.

MACHEN SIE MIT!

MACHEN SIE MIT!
MACHEN SIE MIT!
MACHEN SIE MIT!
MACHEN SIE MIT!

Wollen Sie mithelfen, die nächsten 64'er-Sonderhefte optimal nach Ihren Wünschen zu gestalten? Dann bitten wir Sie, die folgenden Fragen kritisch zu beantworten. Als Bonbon verlosen wir zehn Sonderhefte Ihrer Wahl (mit Diskette). Trennen Sie die Seite aus dem Heft, und schicken Sie sie in einem frankierten Briefumschlag an folgende Adresse (Drucksache genügt, Absender ist nur erforderlich, wenn Sie bei der Verlosung mitmachen).

Markt & Technik Verlag AG
Redaktion 64'er-Sonderhefte
Stichwort: Mitmach-Aktion
Hans-Pinsel-Str. 2
8013 Haar b. München

Wenn ich gewinne
wünsche ich mir Sonderheft Nr. ________

Meine Adresse:

Name, Vorname

____

Straße, Nr.

____

PLZ, Ort

____

Wie alt sind Sie? ________ Jahre

AUSSCHNEIDEN UND EINSENDEN

Die Artikel in diesem Heft sind für mich:

Artikel	leicht verständlich	durchschnittlich	schlecht verständlich
Rosarote Brille	o	o	o
Geometrisches Heimkino	o	o	o
Schildkröte als Sprinter	o	o	o
Bizarre Computerlandschaft	o	o	o
Verschnörkelt oder cool	o	o	o
Spritzige Sprites	o	o	o
Mooving Pictures	o	o	o
In neuem Gewand	o	o	o
Premiere für hohe Auflösung	o	o	o
Effekthascherei	o	o	o
Louvre im C 64	o	o	o
Praktische Tiefkühlkost	o	o	o
Tiefstapler mit Pfiff	o	o	o

Die Artikel in diesem Heft interessieren mich:

Artikel	sehr	durchschnittlich	nicht
Rosarote Brille	o	o	o
Geometrisches Heimkino	o	o	o
Schildkröte als Sprinter	o	o	o
Bizarre Computerlandschaft	o	o	o
Verschnörkelt oder cool	o	o	o
Spritzige Sprites	o	o	o
Mooving Pictures	o	o	o
In neuem Gewand	o	o	o
Premiere für hohe Auflösung	o	o	o
Effekthascherei	o	o	o
Louvre im C 64	o	o	o
Praktische Tiefkühlkost	o	o	o
Tiefstapler mit Pfiff	o	o	o

Die Programme auf Diskette sind für mich:

Programm	maßgeschneidert	brauchbar	uninteressant
Disklader	o	o	o
Interlace	o	o	o
Blitter	o	o	o
Turtle-Grafik	o	o	o
Superfrac	o	o	o
Charmaster	o	o	o
Spritemon 2.2	o	o	o
Sprite-Moover	o	o	o
Profi-Grafik	o	o	o
Hi-Eddi	o	o	o
Sir-Freeze	o	o	o
Tips & Tools	o	o	o

Für die nächsten 64'er-Sonderhefte interessieren mich folgende Themen:

Thema	sehr	durchschnittlich	nicht
Spiele	o	o	o
Grafik	o	o	o
Anwendungen	o	o	o
Tips & Tricks	o	o	o
Hardware	o	o	o
Basic	o	o	o
Assembler	o	o	o
Programmieren	o	o	o
Floppy	o	o	o
Drucker	o	o	o
Sound	o	o	o
Einsteiger	o	o	o

Ich bin Abonennt der 64'er

Ja o   Nein o

Die 64'er-Sonderhefte kaufe ich:

alle o   einige o   keine, ich lese mit o

Mein Wissen über den C 64 schätze ich ein als:

Anfänger o   Fortgeschrittener o   Profi o

Ich besitze folgende Geräte:

älteren C 64 ______ **C64 II** ______

C 128 ______

C 128 D (Blechgehäuse) ______

1541 alt ______ **1541 C** ______

1541-II ______ **1570** ______

1571 ______ **1581** ______

Datasette ______

Drucker ______

An Software benutze ich:

______

______

Zusätzlich habe ich die (den) Computer:

PC ______ Amiga ______ Atari ST ______

andere ______

Die Auslosung der Gewinner erfolgt unter Ausschluß des Rechtsweges. Einsendeschluß ist der 31. 4. 1992, das Datum des Poststempels ist gültig.

Disklader – Programme laden mit Komfort

Diskettenoberfläche de Luxe

Keine umständlichen Ladeanweisungen und ein übersichtliches Inhaltsverzeichnis der Diskette auf dem Bildschirm. Unser »Disklader« erfüllt auch gehobene Ansprüche.

von Herbert Großer

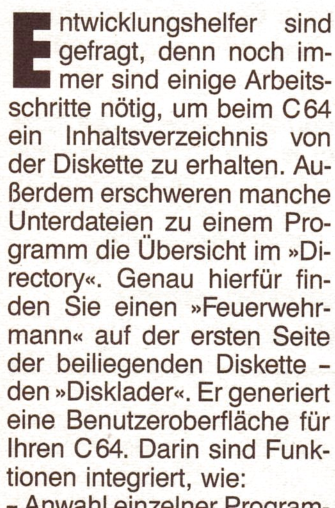
Entwicklungshelfer sind gefragt, denn noch immer sind einige Arbeitsschritte nötig, um beim C64 ein Inhaltsverzeichnis von der Diskette zu erhalten. Außerdem erschweren manche Unterdateien zu einem Programm die Übersicht im »Directory«. Genau hierfür finden Sie einen »Feuerwehrmann« auf der ersten Seite der beiliegenden Diskette – den »Disklader«. Er generiert eine Benutzeroberfläche für Ihren C64. Darin sind Funktionen integriert, wie:

- Anwahl einzelner Programme (mit jeweiliger Kurzbeschreibung),
- automatisches Laden und Starten von Diskette oder
- Erkennung der richtigen Diskette bzw. Diskettenseite.

Da sich der Disklader an erster Stelle auf der Diskette zum Sonderheft befindet, genügt es, zum Laden einzugeben:

```
LOAD":*",8
```

Nach der Bestätigung mit <RETURN> dauert es ca. 15 s, bis die Datei im Speicher ist. Sie starten mit RUN und <RETURN>. Anschließend wird das File entpackt (ca.2 s) und es erscheint die Benutzeroberfläche des »Disklader« (s. Abbildung). In der rechten unteren Bildschirmhälfte sehen Sie weiß umrandet den Namen des ausgewählten Programms. Die unterste Bildschirmzeile ist die dazugehörige Kurzerklärung. Zusätzlich finden Sie in der rechten unteren Bildschirmhälfte den Text »Seite 1 auf Disk« oder »Seite 2 auf Disk«. Da Sie die Inhaltsverzeichnisse beider Seiten (ohne die Disk zu wenden) durchblättern können, finden Sie hier den Hinweis, auf welcher Diskettenseite sich das gewählte Programm befindet.



Durch Tastendruck <CRSR aufwärts> bzw. <CRSR abwärts> wählen Sie das nächste oder vorherige Programm. Sie blättern quasi durch den Inhalt der Programme. <HOME> bringt Sie zum ersten Eintrag des Inhaltsverzeichnisses. Selbstverständlich sind nur die Programme verzeichnet, die sich eigenständig laden oder starten lassen.

<RETURN> führt Sie in den Ladeteil. Ist kein Diskettenfehler aufgetreten, erscheint kurzzeitig »00,OK, 00,00« am Bildschirm. Eventuelle Fehleranzeigen bleiben sichtbar am Bildschirm (z.B. »21,READ ERROR, 18,00«= Drive not ready). Sie lassen sich durch einen beliebigen Tastendruck wieder löschen. Schlagen Sie bitte vorher im Handbuch Ihrer Floppy nach und beseitigen Sie den Fehler. Eine andere Art der Fehlermeldung wird durch einen blinkenden Text dargestellt (z.B. »Bitte Disk wenden« oder »Falsche Diskette«). Sind Fehler ausgeblieben, lädt der Disklader das von Ihnen gewählte Programm von der Diskette und startet es. Ladefehler, die in dieser Phase auftreten, werden nicht mehr berücksichtigt: Der Disklader wird vom neuen Programm einfach überschrieben. Sonst könnten wir nur Programme veröffentlichen, die mit der Benutzeroberfläche zusammenarbeiten. Bei vielen Spielen, Tricks oder Tools ist dies aber nicht der Fall.

Für Sie bedeutet dies, nach jedem Starten eines Programms den »Disklader« erneut zu laden. Wer die Benutzeroberfläche verlassen will, gibt <RUN/STOP> ein. Sie befinden sich dann im normalen »Basic« des C64. Für einen Neustart befehlen Sie

```
SYS 12032
```

und bestätigen mit <RETURN>. Dieser Neustart funktioniert auch nach einem Reset, d.h. wenn Sie durch den entsprechenden Taster einen Hardware-Reset ausgelöst haben. Allerdings sollten Sie zwischenzeitlich kein Programm geladen haben, da dies den verwendeten Speicherbereich überschreiben könnte. Laden Sie in diesem Falle den Disklader neu.

Wir haben bei der Programmierung größten Wert auf Kompatibilität mit den unterschiedlichsten Betriebssystemerweiterungen gelegt. Lediglich bei der Gerätekonfiguration C128 mit RAM-Erweiterung und zweiter Diskettenstation sollten Sie die externe Floppy ausschalten. (gr)

Kurzinfo: Disklader

Programmart: Hilfsprogramm zum Laden der Programme auf der beiliegenden Diskette
Laden: LOAD":*",8
Starten: nach dem Laden mit RUN
Steuerung: Tastatur
Programmautor: H. Großer



Mit Sprites lassen sich die tollsten Effekte zaubern. Unser Editor gehört daher zur Grundausstattung jedes Programmieres.

Der besondere Pfiff von »Spritemon 2.2« liegt in seiner Kürze von nur 14 Blöcken ($0801 bis $1525). Damit läßt sich fast der gesamte Speicher bearbeiten. Trotzdem enthält das Miniprogramm alle wichtigen Funktionen, wie hochauflösend oder Multicolor, Spiegeln, Verschieben und Kopieren. Und als Bonbon lassen sich Bewegungsabläufe entwickeln und ihre Reihenfolge festlegen. Ganze Animations-Sequenzen können dadurch sofort getestet werden. Einige Funktionen lassen sich komfortabel mit dem Joystick in Port 2 realisieren. Haben Sie keinen, tun es die Cursortasten genauso. Geladen wird mit

```
LOAD"SPRITEMON 2.2",8
```

und gestartet mit RUN. Anschließend sehen Sie die Benutzeroberfläche am Bildschirm. Generell gibt's zwei Modi:

Search-Modus

... erlaubt das Durchsuchen des gesamten Speichers nach Sprites. Dargestellt werden dabei jeweils 63 Bytes, als Spritemuster (rechts unten s. Abb.) im Speicherbereich unter »Adress«.

Editor-Modus

... wird mit <RETURN> angewählt. Hier läßt sich ein Sprite gestalten. Ein weiteres <RETURN> erlaubt es, das bearbeitete Sprite an eine beliebige Speicherposition zu kopieren.

Der aktuelle Modus wird im Infofeld (rechts unter dem Sprite, Abb.) gezeigt.

Allgemeine Befehle im Search-Modus

Im Gegensatz zum Editor dient Search hauptsächlich zum Lokalisieren und Sichtbarmachen von Sprites. Einige Befehle erlauben zusätzlich das Kopieren oder Verschieben. Die aktuelle Speicherposition wird dabei unter »Adress« in hexadezimaler Schreibweise gezeigt. Natürlich lassen sich die Sprites (kleines Sprite-Fenster rechts unten) auch horizontal und vertikal vergrößern oder verkleinern

<**CRSR abwärts**>

... addiert zur momentanen Speicherposition den Step-Wert (s. <S>) und macht die dort liegenden 63 Bytes, sichtbar.

<**CRSR rechts**>

... subtrahiert den Step-Wert (s. <S>) von der momentanen Speicherposition. Die 63 Bytes, ab der so berechneten Speicherstelle werden gezeigt.

<**S**>

... schaltet die unter subtrahiert den Step-Wert (s. <S>) von um zwischen Stepweite 0 oder 64. Dieser Wert wird als Schrittweite beim Durchforsten des Speichers verwendet (s. <CRSR abwärts> und <CRSR rechts>).

<**A**>

... erlaubt die Eingabe der Startadresse (links unter »Adress«). Beachten Sie, daß sich nur die Startadresse ändern läßt. Die Endadresse wird automatisch berechnet.

<**C**>

... kopiert das momentan sichtbare Sprite (im kleinen Feld rechts unten) nach Eingabe der Adresse an diese Position. Anschließend wird diese Speicherposition sichtbar gemacht. Wenn Sie wieder zurück zum Ursprungsort wollen, müssen Sie mit <A> diese Speicherposition anwählen.

<**X**>

... wechselt die Breite des Sprites (breit/schmal). Diese Option ist auch vom Editor aus erreichbar.

<**Y**>

... wechselt die Höhe des Sprites (breit/schmal). Auch diese Option ist vom Editor aus erreichbar.

Wenn Sie mit Schrittweite »64« den Speicher durchsuchen, wird es vorkommen, daß jeweils nur Teile von Sprites sichtbar werden. Schalten Sie dann auf »0« zurück und »justieren« Sie mit den Cursortasten das Sprite. Anschließend schalten Sie zurück auf »64«. Da dieser Wert die Länge eines Spriteblocks ist (inkl. einem ungenutzten Byte), lassen sich die darauf folgenden Sprites durchtasten. Einfacher geht es natürlich, wenn Sie die Position der Sprites kennen. Dann brauchen Sie lediglich mit <A> die aktuelle Adresse einzustellen.

Eigene Maschinenprogramme (Search-Modus)

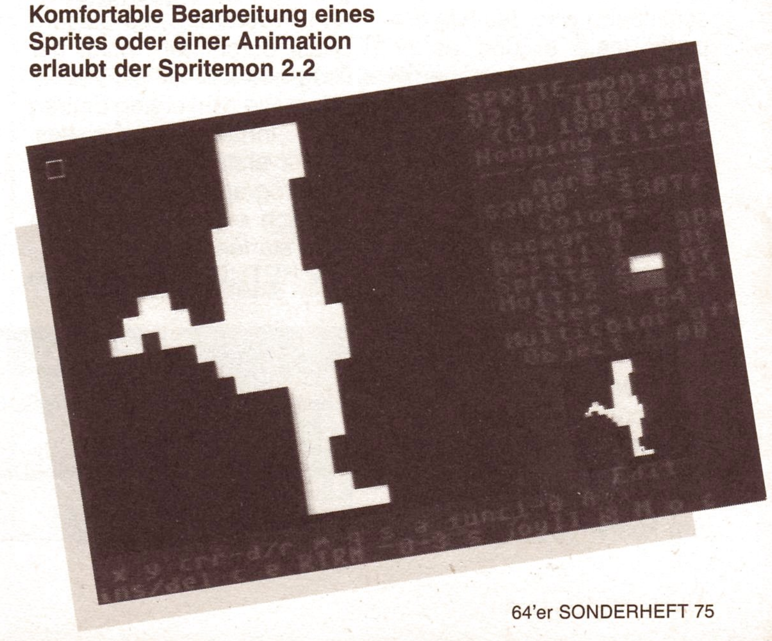
Für eigene Erweiterungsprogramme gibt es einen speziellen Befehl. <**E**> (Extension) ruft ein entsprechendes Maschinenprogramm auf. Weil danach auf die Eingabe der Einsprungadresse gewartet wird, lassen sich auch unterschiedliche Programme im Speicher anspringen. Beachten Sie, daß Ihre Routinen mit »RTS« abgeschlossen sein müssen. Dann erfolgt automatisch ein Rücksprung zum Spritemonitor. Außerdem darf Ihr Programm den Speicherbereich $02A7 bis $02AE nicht verändern und selbstverständlich nicht das Programm des Spritemon ($0801 bis $14E6).

Die Funktion Extension kann während der Eingabe mit <RUN/STOP> abgebrochen werden.

Animationen im Spritemon (Search-Modus)

Im Gegensatz zu den meisten anderen Editoren ist es Spritemon egal, an welcher Speicherposition und in welcher Anord-

Komfortable Bearbeitung eines Sprites oder einer Animation erlaubt der Spritemon 2.2



nung Animations-Sprites liegen (bis max. 99). Die Reihenfolge läßt sich beliebig definieren. Es muß lediglich darauf geachtet werden, daß kein Sprite zweimal in einer Sequenz vorkommt. Bevor Sie eine Animation betrachten, müssen Sie daher Ihre Reihenfolge festlegen. Dafür existiert die Funktion Markieren (<SHIFT INS/DEL>). Die Animationsnummer wird dabei angezeigt (jedesmal wenn Sie den Sprite-Block anwählen). Ein verkehrt zugeteilter Block läßt sich löschen (<INS/DEL>) und falls Sie eine andere Reihenfolge wünschen wählen Sie einfach Gesamtlöschen (<SHIFT CLR/HOME>). Im Anschluß an die Reihenfolge wählen Sie <SHIFT A> und befinden sich im Animationsmenü. Die Animation selbst kann mit der Leertaste oder durch Druck auf den Feuerknopf (Port 2) gestartet werden. Gestoppt wird genauso. Während des Ablaufs verlangsamen oder beschleunigen Sie die Bilderfolge (0 bis 32) über die Cursortasten (oder Joystick rechts/links). <RETURN> bricht die Animation ab.

<SHIFT INS/DEL>
... markiert einen Spriteblock als Animationsbild. Welche Nummer er in der Animationsreihenfolge hat, wird neben »Objekt« angezeigt.

<INS/DEL>
... löscht einen Spriteblock aus der Animation.

<SHIFT CLR/HOME>
... löscht den gesamten Animationsspeicher.

<SHIFT A>
... führt ins Animationsmenü.

<SPACE>
... startet (oder stoppt) eine Animation am Bildschirm.

<CRSR rechts> oder **Joystick rechts**
... drosselt die Geschwindigkeit.

<CRSR links> oder **Joystick links**
... erhöht die Geschwindigkeit.

<RETURN>
... beendet das Animationsmenü.

Multicolor und Spritefarben (Search- und Editor-Modus)

Wenn Sie ein Muster aus vertikalen Linien entdecken, handelt es sich sicherlich um ein Mehrfarbensprite. Spritemon läßt sich für diesen Fall vom Einfarbenmodus auf Multicolor umschalten. Dann kann ein Sprite vier Farben beinhalten. Diese sind einzeln über die Funktionstasten einstellbar. Eine Betätigung der u.a. Tasten schaltet jeweils die entsprechende Farbe aufwärts oder abwärts, entsprechend der Farbnummer (s. Handbuch). Beachten Sie, daß im Einfarbenmodus nur zwei Farben wirksam sind (Background und Sprite).

<M>
... erzeugt wechselweises Umschalten zwischen Single- und Multicolor.

<F1>
... erhöht den Farbwert der Hintergrundfarbe (Backgr.).

<F2>
... senkt den Farbwert der Hintergrundfarbe (Backgr.).

<F3>
... erhöht den Farbwert der Multicolorfarbe 1 (Multi1).

<F4>
... mindert den Farbwert der Multicolorfarbe 1 (Multi1).

<F5>
... erhöht den Farbwert der Spritefarbe (Sprite).

<F6>
... erniedrigt den Farbwert der Spritefarbe (Sprite).

<F7>
... erhöht den Farbwert der Multicolorfarbe 2 (Multi2).

<F8>
... senkt den Farbwert der Multicolorfarbe 2 (Multi2).

Funktionen im Editor

Der Zeichencursor wird über die Cursortasten oder mit dem Joystick in Port 2 bewegt. Im Editor sind Funktionen eingebaut wie Spiegeln, Mischen mit einem anderen Sprite, Srollen in alle Richtungen und last but not least die Änderung der Sprite-Farben.

<X>
... wechselt die Breite des Sprites (breit/schmal).

<C>
... wechselt die Höhe des Sprites (breit/schmal).

<M>
... bewirkt Umschalten zwischen Single- und Multicolor.

<0> (Null)
... der Hintergrund (Backgr) ist Zeichenfarbe.

<1>
... Multicolor 1 (Multi1) ist Zeichenfarbe (nur bei Multicolor).

<2>
... die Spritefarbe (Sprite) ist Zeichenfarbe.

Kurzinfo: Spritemon 2.2

Programmart: Sprite-Editor
Laden: LOAD"SPRITEMON 2.2",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 14
Programmautor: H. Eilers

<3>
... Multicolor 2 (Multi2) ist Zeichenfarbe (nur bei Multicolor).

<SPACE> oder **Joystick Feuer**
... setzt einen Punkt in der gewählten Farbe.

<CRSR hoch> oder **Joystick hoch**
... Cursor nach oben.

<CRSR runter> oder **Joystick runter**
... Cursor nach unten.

<CRSR rechts> oder **Joystick rechts**
... Cursor nach rechts.

<CRSR links> oder **Joystick links**
... Cursor nach links.

<SHIFT CLR/HOME>
... löscht das Editorfeld.

<S>
... schaltet in den Scroll-Modus. Hier scrollt der Editorinhalt in die Richtung des Joysticks oder der Cursortasten. <SPACE> führt zurück zum Editor.

<SHIFT M>
... schaltet in den Spiegelmodus. Hier wird der Editorinhalt in die Richtung des Joysticks oder der Cursortasten gespiegelt. <SPACE> führt zurück zum Editor.

<O>
... mischt den Editor-Sprite mit einem aus dem Speicher. Dann erwartet Spritemon die Eingabe der Speicherposition.

<C>
... copiert den Editor-Sprite an eine beliebige Position. Danach erwartet Spritemon die Eingabe der Speicherposition.

<RETURN>
... schaltet zurück zum Search-Modus. Vorher muß allerdings die Speicherposition angegeben werden.

<Q>
... beendet Spritemon 2.2 ohne Sicherheitsabfrage.

Da das Programm keine Lade- oder Speicherfunktionen eingebaut hat, benötigen Sie dazu ein Hilfsprogramm. Ab Seite 47 finden Sie die Beschreibung eines solchen. Notieren Sie sich die Position Ihrer Sprite-Animation, laden Sie

```
LOAD"S-SUCHER V2.0",8
```

und starten Sie mit RUN. Mit diesem Programm läßt sich die Animation speichern. (gr)

Animation ist eines der faszinierensten Themen in der Computergrafik. Anders als bei den meisten Computern wird ihre Programmierung beim C64 erleichtert – durch kleine grafische Objekte, Sprites genannt. Sie lassen sich ohne großen Rechenaufwand frei über den Bildschirm verschieben. Die Daten erhalten sie aus einem definierten Speicherbereich. Aus welchem, läßt sich durch Ändern einer einzigen Speicherstelle festlegen. Das Video-Interface-Chip (VIC) des C64 kann (ohne Tricks) bis zu acht Stück gleichzeitig darstellen. Doch bevor eines dieser Sprites zu laufen beginnt, müssen Bewegungsabläufe konstruiert werden. Diese Animationssequenzen bestehen aus den Daten für mehrere Einzel-Sprites, die (wie beim Film) einzelne eingefrorene Bewegungen enthalten. Wenn diese schnell nacheinander auf dem Bildschirm dargestellt werden, läßt sich das Auge täuschen, und es entsteht der Eindruck einer fließenden Bewegung. Der Spritemon in diesem Heft (ab S. 24) erlaubt die Konstruktion von Sprites und Bewegungssequenzen. Die einzelnen Bewegungsabläufe lassen sich sogar testen. Unser Sprite-Moover versetzt Sie in die Lage, eine Sequenz dann eigenständig ablaufen zu lassen. Die fließende Bewegung im Bild ist nämlich komplizierter zu programmieren als man zunächst meinen könnte:

Nehmen wir an, Sie programmieren einen ständigen Wechsel der einzelnen Spritepointer. Dann haben Sie zwar Ihre Sequenz animiert, es wird aber ein zyklisches Flackern am Bildschirm erscheinen. Der Grund dafür ist eine Interferenz zwischen Bildwechselfrequenz des Video-Interface-Chips (VIC) und Animationsgeschwindigkeit. Mit anderen Worten: Da das Bild vom VIC zeilenweise aufgebaut wird, kommt es vor, daß ein Teil des ersten Sprites schon am Bildschirm ist, danach wechselt der Sprite-Pointer, und in den nächsten Zeilen wird der untere Teil des neuen Sprites gezeigt. Dieser Effekt führt zu einem unnatürlichen Flackern.

Kurzinfo: Sprite-Moover

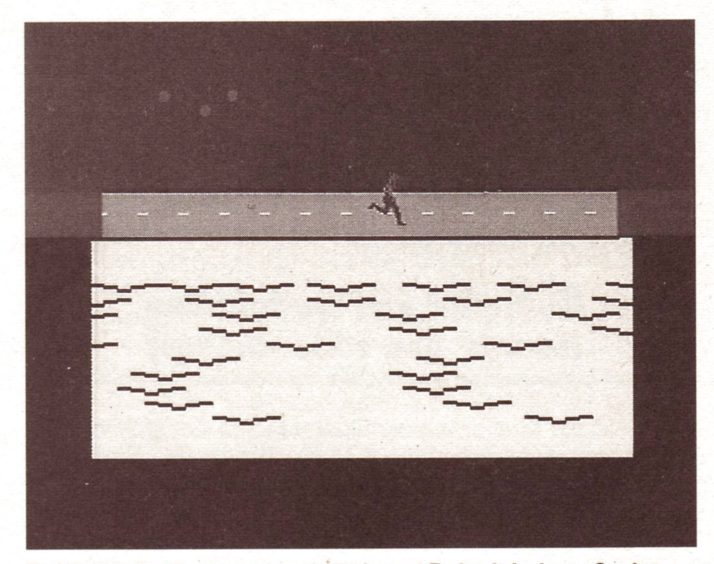
Programmart: Animationsprogramm für Sprites
Laden: LOAD"SPRITE-MOOVER",8,1
Starten: SYS49152 oder SYS49152+3
Steuerung: über POKEs im Direktmodus oder Programm (s. Textkasten)
Besonderheiten: läuft im Interrupt ab
Benötigte Blocks: 11
Programmautor: H. Großer

Kurzinfo: Demo Moover

Programmart: Demo für Sprite-Moover
Laden: LOAD"DEMO.MOOVER",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 15
Programmautor: H. Großer

Bildwechsel im Interrupt

Abhilfe schafft die Einbindung in einen Rasterinterrupt. Die Umschaltung wird dabei mit dem Bildwechsel synchronisiert und damit kann kein Flackern mehr stattfinden. Achtung: Die Ausführungszeit der Routine muß so kurz wie nur irgendwie möglich sein. Je näher man an einen Bildwechsel kommt, desto kürzer wird die Zeit zwischen den Interrupts. Gerade hier wird aber das eigentliche Programm erledigt. Folge: Es wird u. U. extrem langsam. Ein Trick zur Überwachung dieser Ausführungszeiten ist ein Umschalten zu Beginn des Interrupts auf die Rahmenfarbe, am Ende dann wieder auf den Ursprungswert. Dadurch wird am Bildschirmrand ein Streifen sichtbar, der zum Zeitbedarf des Interrupts proportional ist. Mit dieser Methode läßt sich ein Rasterzeilen-IRQ auch so postionieren, daß er die Daten dann ändert, wenn keine Sprites am Bildschirm gezeichnet werden. Ansonsten wären wieder Störungen sichtbar. Sie werden allerdings beim Farbstreifen



Realistische Bewegungsabläufe am Beispiel eines Sprinters sehen Sie im Demo

unterschiedliche Breiten feststellen. Vermeiden kann man dies nur, wenn alle IRQ-Abschnitte gleiche Ausführungszeiten haben. Damit Sie das Rad aber nicht immer wieder erfinden (programmieren) müssen, ist diese Funktion bei Sprite-Moover schon eingebaut. Da es sich anbietet, wenn schon im Interrupt gearbeitet wird, mehrere Funktionen einzubinden, läuft die komplette Routine im Interrupt ab. Für Sie bedeutet dies, Sie können zusätzlich ein Basic-Programm laufen lassen. Bewegung, Animation, und sogar ein begrenztes Hintergrund-Scrolling lassen sich durch POKEn in bestimmte Speicherstellen beeinflussen (s. Textkasten). Zum Ausprobieren ist ein Demoprogramm mit auf der beiliegenden Diskette. Tippen Sie dazu ein:

```
LOAD"DEMO.MOOVER",8
```

und starten Sie danach mit RUN. Ein Läufer rennt über eine Straße (Abb.). Wie Sie die einzelnen Parameter ändern können, ersehen Sie aus dem Textkasten. Ebenso die Speicherpositionen, an denen sich die Animations-Sprites bei eigenen Kreationen befinden müssen. Der Bereich dafür ist immer von Speicherstelle 12288 ($3000) bis 14079 ($36FF) festgelegt. (gr)

Animationsdaten von Sprite-Moover

Geladen wird die Routine mit:
LOAD"SPRITE-MOOVER",8,1
Ein Start kann auf zwei Arten erfolgen:
1. SYS49152 – löscht den Bildschirm und startet.
2. SYS49152+3 – startet ohne Bildschirmlöschen (wird im Demo verwendet)
Ein Animationsbild setzt sich aus vier Sprites zusammen, von denen je drei zugleich dargestellt werden. Sie stehen bei den Demo-Sprites in der Reihenfolge
1. Haar,
2. Beine und Rumpf,
3. Arme und Gesicht (Phase 1) und
4. Arme und Gesicht (Phase 2)
im Speicher. Für die Bewegungssequenzen der Arme (und Gesicht) werden zwei Phasen verwendet. Damit erstreckt sich eine Armsequenz über zwei Bein-Haar-Sequenzen. Die Sprites müssen ab Position 12288 ($3000) bis 14079 ($36FF) und besteht damit aus sieben Phasen für Haar, Beine und Rumpf und aus 14 Phasen für Arme und Gesicht.

Für eigene Konstruktionen läßt sich vor dem Start der Routine (mit SYS49152) die horizontale Position ändern. Dies geschieht durch POKEn der neuen Werte in die beschriebenen Speicherstellen (z.B. setzt »POKE49204,63« die Haare um 20 Zeilen höher):
49204 – Haar (Normalwert = 83)
49205 – Beine und Rumpf (Normalwert = 95)
49206 – Arme und Gesicht, beide Phasen (Normalwert = 104)

Ebenso lassen sich vor dem Start die Farben der Sprites festlegen:
49620 – Haar (Normalwert 9 = Braun)
49625 – Beine und Rumpf (Normalwert 6 = Blau)
49630 – Arme und Gesicht, beide Phasen (Normalwert 8 = Orange)

Für die Hintergrundfarben des Bildschirms und für die Rahmenfarben stehen sechs Speicherstellen zur Verfügung:
49198 – oberer Bildschirmrand
49199 – mittlerer Streifen, seine Bildschirmzeichen werden gescrollt
49200 – unterer Bildschirmrand
49201 – Rahmen oben
49202 – Rahmen neben dem gescrollten Bereich
49203 – Rahmen unten

Die Werte aus diesen Speicherstellen werden als Farbcodes interpretiert (s. Handbuch):
POKE49198,1
ändert die Hintergrundfarbe des oberen Rand auf Weiß,
POKE49198,0
färbt diesen Hintergrundstreifen schwarz.

Die Abläufe der Bewegungen können vielfältig beeinflußt werden. Da das komplette Programm im Interrupt abläuft, quasi parallel zu einem Basic-Programm, lassen sich diese Veränderungen durch direkte Eingabe testen. Eine der Speicherstellen ist für den automatischen Ablauf zuständig:
POKE49194,1
schaltet auf automatische Frequenz um. Hier werden die Geschwindigkeit der Animation, die Bewegung über den Bildschirm und die Geschwindigkeit des Hintergrundscrollens aus einer Tabelle gelesen (s. u.). Die Zeit (Anzahl der Bildwechsel bis zu den nächsten Tabellenwerten) ist dabei in Speicherstelle 49195 festgelegt. Kurze Werte ergeben schnelle Tabellenwechsel, lange Werte langsamere. Mit
POKE49194,1
ist die Automatik ausgeschaltet. Dann lassen sich die einzelnen Parameter direkt beeinflussen. Zeitbestimmende Werte beziehen sich dabei auf den Bildwechsel. D.h. da jede fünfundzwanzigstel Sekunde ein Bildwechsel stattfindet und dabei die Auswertungsroutine aktiviert wird, berechnet sich die Zeit:
Wert/25 = Zeit in Sekunden

Da für jede Speicherstelle die Werte 0 bis 255 erlaubt sind, ist dies eine Zeit zwischen 1/25 Sekunde bis ca. 1 Sekunde (250).

49190 – Zeit nach denen die Sprites nach rechts geschoben werden. Um wieviel Bildschirmpixel geschoben wird, ist in 49152 festgelegt.

49191 – Zeit nach der jeweils die Animatition (=Spritepointer) gewechselt wird.

49192 – Anzahl der Pixel um die nach rechts geschoben wird. Die Zeit nach der dies geschieht ist in 49190 festgelegt. Obwohl auch hier 0 bis 255 erlaubt ist, sind kleinere Werte sinnvoll.

49193 – die Zeit nach der der mittlere Bildschirmstreifen um zwei Bildpunkte nach links scrollt.

Für die Automatikfunktion existiert eine Tabelle, bestehend aus 4 x 16 Werten:

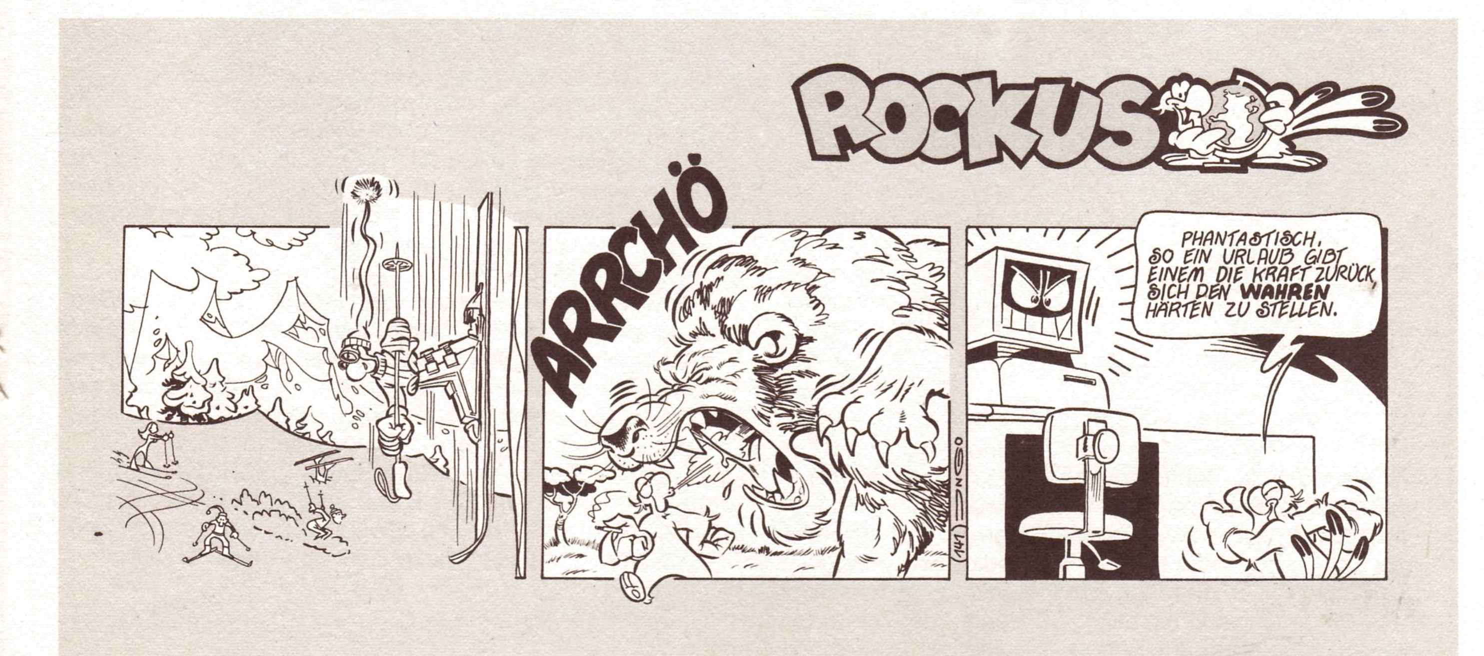
49215 bis 49230 – Tabelle der Zeit nach denen die Sprites nach rechts geschoben werden. Um wieviel Bildschirmpixel ist dabei in der Tabelle ab 49247 festgelegt.

49231 bis 49246 – Tabelle der Zeit nach der jeweils die Animation (=Spritepointer) gewechselt werden.

49247 bis 49262 – gibt jeweils die Anzahl der Pixel an, um die nach rechts geschoben wird. Die Zeit nach der dies geschieht ist ab 49215 festgelegt.

49263 bis 49278 – Zeittabelle nach der der mittlere Bildschirmstreifen um zwei Bildpunkte nach links scrollt.

Die Tabelle wird (wenn Automatik eingeschaltet, s. 49194) für die Bewegungssequenzen verwendet. Das heißt: zuerst werden jeweils die Werte aus 49215, 49231, 49247 und 49263 in die Speicherstellen 49190, 49191, 49192 und 49193 übertragen. Die Zeit, die eine Wertereihe gehalten wird, ist dabei in Speicherstelle 49195 festgelegt. Nach Ablauf der Zeit (49195) werden automatisch 49216, 49232, 49248 und 49264 übertragen und wieder die festgelegte Zeit gehalten. Nach den letzten Tabellenwerten startet die Automatik wieder bei den ersten.



Zeichensatz – drei Modi

IN NEUEM GEWAND

Der C64 erlaubt Änderungen seiner Schriftzeichen. Die Anwendungen reichen dabei vom Text mit deutschen Umlauten über Hieroglyphen bis zu mehrfarbigen Spielelandschaften. Sehen Sie selbst, wie einfach solche Manipulationen sind.

Der Video-Chip des C64 ist einer der komfortabelsten Grafik-ICs, obwohl so mancher Besitzer eines anderen Computers das nicht wahrhaben will. Die Grafikauflösung wird von VGA-Karten in Personalcomputern (PC) zwar übertroffen und in puncto Farbenvielfalt hat der Amiga seine Nase vorn. Aber spätestens wenn es um Änderung von Textzeichen geht, gibt es keinen Rechner, bei dem dies mit so wenig Aufwand und ohne Geschwindigkeitsverlust zu erreichen ist.

Durch Ändern der Register 53248 ($D000) bis 53294 ($D02E), läßt sich das Aussehen der Zeichen von dem eingebauten Festwertspeicher (ROM) auf einen von 32 Bereichen des normalen Datenspeichers (RAM) umschalten (Tabelle 3). Hier sollte allerdings das Aussehen der gewünschten Zeichen vorher festgelegt sein. Jeweils eines davon besteht immer aus einer Matrix von 8 x 8 Punkten. Genauer acht Zeilen mit je acht Punkten Breite. Da ein Byte 8 Bit beinhaltet, läßt sich jeweils eine Zeile in einem Byte codieren. Ein gesetztes Bit bedeutet dabei einen Punkt auf dem Bildschirm (Abb. 1). 40 dieser Zeichen passen nebeneinander auf einen Textbildschirm und 25 untereinander. Damit passen 1000 Zeichen auf einen Textbildschirm. Doch wie arbeitet der VIC:

Er benötigt dazu einen Bildschirmspeicher mit der Länge von 1000 Zeichen (nach dem Einschalten ab Adresse 1024 ($0400) bis 2023). Jede der Speicherstellen steht zwar für einen Buchstaben am Bildschirm, kann aber nur 1 Byte (0 bis 255) beinhalten. Das genügt natürlich nicht, um das Aussehen des Zeichens zu speichern. Aber wenn wir einfach zur Kenntnis nehmen, daß insgesamt 256 Zeichenmuster in einem Zeichensatz zur Verfügung stehen, bezeichnet die Zahl eines davon in dem vorher angesprochenen Zeichensatzspeicher. Sie brauchen also lediglich einen Wert in den Bildschirmspeicher zu schreiben, den Rest erledigt der VIC. Er holt sich das Aussehen dieses Zeichens aus dem Zeichensatzspeicher und am Bildschirm erscheint z.B. ein Buchstabe. Dabei ist es egal, an welchem der 32 Bereiche ein Zeichensatz angesiedelt ist, man muß es dem VIC nur vorher mitteilen. Wenn Sie also jetzt folgendes eingeben:

```
POKE 1024,0
```

werden Sie sich nicht wundern, wenn »@« am linken oberen Eck des Bildschirms erscheint. Diese Bildschirmcodes sollten Sie im Handbuch nachlesen, denn sie stimmen nicht mit den ASCII-Codes überein. Der Grund: Auf die speziellen Codes reagiert der Computer, er braucht sie nicht darzustellen. Und falls doch, verwendet er die zweite Hälfte der Zeichenmuster, denn hier sind die reversen Muster der ersten 128 Zeichen gespeichert. Sie sehen diese immer, wenn Sie den Cursor über ein Zeichen bewegen. Dann blinkt dieses Zeichen.

Die Position des Bildschirmspeichers läßt sich ebenfalls verändern, da dieser Speicher aber nur 1000 Zeichen lang ist und der C64, wie der Name auch sagt, 64 KByte Speicher besitzt. in 64 Stufen (Tabellen 1 und 2). Jetzt wird auch klar, warum sich der Zeichensatz selbst in 32 Stufen verlegen läßt: 8 Byte x 256 Zeichen = 2048 Bytes (2 KByte).

Ein dritter Speicher ist indirekt noch für das Aussehen der Bildschirmdarstellung verantwortlich – das Farb-RAM. Es läßt sich allerdings nicht verschieben und liegt immer an Position 55296 ($D800) bis 56296 ($DBE7). In ihm wird anhand der Farbnummer (s. Handbuch) auch die Farbe des Zeichens festgelegt. Beachten Sie: Die Farbe des Zeichenhintergrundes ist in Register 53281 festgelegt.

Wahl der Video-Bank

Der Videocontrollerbaustein kann immer nur auf einen Speicherbereich von 16 KByte zugreifen. Da aber 64 KByte RAM untergebracht sind, läßt er sich auf vier verschiedene Banks (oder Abschnitte) umschalten. Die Bankanwahlbits, die einen Zugriff auf die verschieden Speicherabschnitte ermöglichen, befinden sich im Complex-Interface-Adapter 6526, #2 (CIA #2). Mit der Basic-Anweisung POKE und PEEK wird eine Bank durch Steuerung der Bits 0 und 1 von Port A des CIA #2 (56576 oder hex $DD00) gewählt. Zur Änderung der Speicherabschnitte müssen diese 2 Bit auf Ausgabe stehen:

```
POKE 56578, PEEK (56578) OR 3
```

setzt die Bits 0 und 1

```
POKE 56576,(PEEK(56576) AND 252) OR A
```

Belegung	binär	PEEK
	00011000	24
	00111100	60
	01100110	102
	01111110	126
	01100110	102
	01100110	102
	01100110	102
	00000000	0

[1] Der Buchstabe »A« steht im Zeichen-ROM in den Speicherstellen von 53256 bis 53263

Belegung	Bit-Muster
**	00011000
****	00111100
** **	01100110
******	01111110
** **	01100110
** **	01100110
** **	01100110
	00000000

[2] Die Bits des Buchstabens »A« im Normalmodus

Darstellung	Bit-Muster
AABB	00011000
CCCC	00111100
AABBAABB	01100110
AACCCCBB	01111110
AABBAABB	01100110
AABBAABB	01100110
AABBAABB	01100110
	00000000

[3] Die Bits des Buchstabens »A« im Multicolor-Modus

wechselt die VideoBank. »A« muß einen der Werte aus Tabelle 2 haben. Das Konzept der 16-KByte-Abschnitte spielt bei allen Anwendungen des VIC eine Rolle. Sie sollten stets wissen, auf welche Bank der Videocontroller zugreift, denn dadurch wird festgelegt, woher die Zeichendatenmuster kommen und wo sich der Bildschirmspeicher befindet. Nach dem Einschalten des C64 wird automatisch Bank 0 ($0000 - $3FFF) aktiviert.

Achtung: Der originale Zeichensatz des C64 ist in den Banks 1 und 3 für den VIC-Baustein nicht verfügbar.

Bildschirmspeicher

Durch POKEn ins Kontrollregister 53272 ($ D018 HEX) kann die Adresse des Bildschirmspeichers geändert werden. Dieses Register wird jedoch auch zur Steuerung des jeweils benutzten Zeichensatzes verwendet. Achten Sie daher besonders darauf, diesen Teil des Steuerregisters nicht zu stören. Die oberen 4 Bit steuern den Bildschirmspeicher. Verwenden Sie folgende Anweisung, um den Bildschirm in einen anderen Bildbereich zu legen:

```
POKE 53272,(PEEK(53272)AND 15) OR A
```

Die Werte für »A« finden Sie in Tabelle 1.

Farbspeicher

Der Farbspeicher kann nicht verschoben werden. Er befindet sich stets im Adreßbereich von 55296 ($D800) bis 56295 ($DBE7). Bildschirmspeicher und Farbspeicher werden in den verschiedenen Grafikmodi unterschiedlich benutzt.

```
zeichensatz-betrachter

f1 = cbm/standard     f2 = neu laden
f3 = neue zeichen     f4 = programm ende
f5 = multicolor
f7 = extended color

 1 = rahmenfarbe (53280)
 2 = farbe1      (53281)
 3 = farbe2      (53282)
 4 = farbe3      (53283)
 5 = farbe4      (53284)
 6 = zeichenfarbe   0- 63
 7 = zeichenfarbe  64-127
 8 = zeichenfarbe 128-191
 9 = zeichenfarbe 192-255
```

[4] »Betrachter« simuliert alle Textmodi

A	Bits	Lage	
		dezimal	hexadez
0	0000XXXX	0	$0000
16	0001XXXX	1024	$0400 Std
32	0010XXXX	2048	$0800
48	0011XXXX	3072	$0C00
64	0100XXXX	4096	$1000
80	0101XXXX	5120	$1400
96	0110XXXX	6144	$1800
112	0111XXXX	7168	$1C00
128	1000XXXX	8192	$2000
144	1001XXXX	9216	$2400
160	1010XXXX	10240	$2800
176	1011XXXX	11264	$2C00
192	1100XXXX	12288	$3000
208	1101XXXX	13312	$3400
224	1110XXXX	14336	$3800
240	1111XXXX	15360	$3C00

1. Die oberen 4 Bit des Kontrollregisters 53272 bestimmen die Lage des Bildschirmspeichers

Zeichensatzspeicher

Normalerweise erhält der Baustein die Formen der anzuzeigenden Zeichen vom Character-Generator-ROM. In diesem Baustein sind alle Muster gespeichert wie verschiedene Buchstaben, Zeichen und Interpunktionssymbole.

Eines der Merkmale des C64 ist seine Fähigkeit, im RAM-Speicher befindliche Muster zu benutzen. Diese RAM-Muster werden vom Anwender selbst kreiert, so daß ein nahezu unbegrenzter Satz an Symbolen zur Verfügung steht.

Ein normaler Zeichensatz enthält 256 Zeichen, bei dem jedes Zeichen durch 8 Byte bestimmt wird. Da jedes Zeichen also 8 Byte beansprucht, benötigt der komplette Zeichensatz 256 x 8 = 2 KByte (2048 Byte). Da der Videocontrollerbaustein gleichzeitig auf 16 KByte zugreift, gibt es pro Bank acht verschiedene Speicherplatzmöglichkeiten für einen vollständigen Zeichensatz. Sie brauchen natürlich nicht immer einen Zeichensatz zu verwenden. Er muß jedoch stets an einem der acht möglichen Startplätze beginnen. Die Lage des Zeichenspeichers wird durch 3 Bit vom Videocontroller kontrolliert (53272 oder Hex $D018). Die Bits 3, 2 und 1 steuern die Lage des Zeichensatzes, Bit 0 wird überlesen. Um die Lage des Zeichensatzes zu ändern, dient die Basic-Anweisung:

```
POKE 53272,(PEEK(53272)AND240) OR A
```

Entnehmen Sie die Werte Tabelle 3.

Nur wenn der VIC auf die ROM-Zeichendaten zugreift, wird auch das ROM eingeschaltet. Ansonsten wird dieser Bereich von den Ein-/Ausgaberegistern beansprucht. Falls Sie das Zeichen-ROM benötigen, z.B. um ihn ins RAM zu kopieren und seine Zeichen zu verändern, müssen Sie das Ein-/Ausgaberegister aus- und das Zeichen-ROM einschalten. Erst jetzt kann kopiert werden. Danach muß das Ein-/Ausgaberegister erneut eingeschaltet werden. Während des Kopierens (bei ausgeschalteter Ein-/Ausgabe) sind keine Interrupt-Unterbrechungen erlaubt, sonst stürzt der Rechner ab. Zum Abschalten des IRQ müssen Sie den Programmodus verwenden, da ebenfalls die Tastatur ausgeschaltet ist:

```
10 POKE 56334,PEEK(56334)AND 24
```

Anschließend schalten Sie den Ein-/Ausgabe-Bereich aus und das Zeichen-ROM ein:

```
20 POKE 1, PEEK (1)AND 251
```

Das Zeichen-ROM befindet sich im Bereich von 53248 bis 57343 ($D000 bis $DFFF). Sie kopieren es mit

```
30 FORI=OTO 2047: POKE 4096+I,PEEK(53248+I):NEXT
```

Um die Ein-/Ausgabe für den normalen Betrieb zu aktivieren, ist folgende POKE-Anweisung erforderlich:

```
40 POKE 1,PEEK(1)OR4
```

Danach wird der Interrupt wieder eingeschaltet:

```
50 POKE 56334,PEEK(56334)OR1
```

und der Zeichensatz aktiviert

```
60 POKE 53272,(PEEK(53272)AND240) OR 4
```

Einen Zeichensatz in Basic zu kopieren, ist sehr zeitaufwendig. Um Ihnen einen Vergleich zu geben, befinden sich zwei Programme mit auf der beiliegenden Diskette:

```
LOAD"COPY.Basic",8
```

kopiert nach RUN den Zeichensatz (in Basic) und schaltet auf den neuen um. Es läßt sich auch in eigene Programme einbauen, ebenso wie

```
LOAD"COPY.MASCH",8,1
```

mit dem Unterschied, daß dieses Programm mit SYS49152 gestartet wird und in Maschinensprache geschrieben ist. In Basic-Programme wird es am Anfang nachgeladen z.B. mit:

```
0 IFA=OTHENA=1:LOAD"COPY.MASCH",8,1
```

Aktiviert wird es bei Bedarf (SYS49152). Beachten Sie lediglich, daß Ihr Basic-Programm die Speicherposition 4096 nicht überschreiten darf.

Standardzeichenmodus

Beim Einschalten des C64 befindet sich dieser im Standardzeichenmodus, in dem Sie normalerweise programmieren. Zeichen können aus dem ROM oder dem RAM gelesen

werden. Normalerweise wird jedoch auf die Zeichen im ROM zugegriffen. Benötigen Sie für ein Programm spezielle Grafikzeichen, sind lediglich die neuen Zeichenmuster im RAM zu definieren und der VIC umzuschalten (s. Zeichenspeicher).

Um Zeichen auf dem Bildschirm in Farbe zu zeigen, greift der Video-Controller-Baustein auf den Bildschirmspeicher zu, damit der Zeichencode für diesen Bildschirmplatz bestimmt wird. Gleichzeitig nimmt er aus dem Farbspeicher die Farbe für die Zeichenanzeige.

Um ein Zeichen zu ändern, benötigen Sie zuerst den Bildschirmcode. Er ist mit 8 zu multiplizieren. Danach wird der Beginn des Zeichenspeichers addiert plus Bank-Adresse:

Zeichenadresse = Bildschirmcode x 8 + (Zeichensatz x 2048) + (Bank x 16348)

Zeichendefinition

Jedes Zeichen wird aus einer Matrix von 8 x 8 Punkten gebildet. Hierbei können die einzelnen Punkte entweder ein- oder ausgeschaltet sein. Beim Commodore 64 sind die Zeichenbits im Zeichengenerator ROM abgelegt. Jedes Zeichen ist dabei als Satz von 8 Byte gespeichert. Jedes Byte steht für das Punktmuster einer Reihe im Zeichen und jedes Bit für einen Punkt. Der Zeichenspeicher im ROM beginnt ab 53248 (bei ausgeschalteter Ein-/Ausgabe). Die ersten 8 Byte ab 53248 ($D000) bis 53255 ($D007) enthalten das Muster für das Zeichen »@«, dessen Zeichencodewert im Bildschirmspeicher 0 ist. Die nächsten 8 Byte ab 53265 ($D00F) enthalten die Information zur Bildung des Buchstabens A (Abb.1).

Jeder vollständige Zeichensatz beansprucht eine Speicherkapazität von 2 KByte (2048 Bit). Insgesamt sind 256 Zeichen vorhanden. Da es insgesamt zwei Zeichensätze gibt (Umschaltung mit <CBM SHIFT>), enthält der ROM-Zeichenspeicher 4 KByte.

Programmierbare Zeichen

Alle Buchstaben, Zahlen oder Standardgrafikzeichen vom C 64 müssen zuerst in den RAM-Speicher kopiert werden, damit Sie ihn in Ihrem Programm benutzen können.

Achtung: Achten Sie darauf, daß Ihr Zeichensatz nicht vom Basic-Programm überschrieben wird!

Eine Adresse darf auf keinen Fall als Beginn des Zeichensatzes gewählt werden: Adresse 0. Im C 64 werden ab dieser Position Seite 0 (0-Page) wichtige Daten gespeichert.

Für Ihren Zeichensatz stehen jedoch noch genügend weitere Anfangspositionen zur Verfügung. Am besten wählen Sie hierzu den Bereich ab 12288 ($3000), indem Sie zu den unteren 4 Bit in Speicherzelle 53272 den Wert 12 addieren. Probieren Sie nun folgende POKE-Anweisung aus:

```
POKE 53272,(PEEK(53272)AND240)+12
```

Sofort sind alle Buchstaben vom Bildschirm verschwunden, weil nämlich bis jetzt noch kein Zeichensatz ab Adresse 12288 steht...nur zufällige Bytes. Kehren Sie mit <RUN/STOP RESTORE> wieder zurück in den Normalmodus.

Nun wollen wir Grafikzeichen konstruieren. Um Ihren Zeichensatz zu schützen, sollten Sie die Speicherkapazität für das Basic reduzieren. Der Speicher in Ihrem Computer bleibt unverändert. Sie haben lediglich dem Basic die Anweisung gegeben, einen bestimmten Teil nicht zu benutzen:

```
PRINT FRE(0)-(SGN(FRE(0))<0)*65535
```

Die gezeigte Zahl gibt die unbenutzte Speicherkapazität an. Geben Sie nun folgendes im Direktmodus ein:

```
POKE 52,48:POKE 56,48:CLR
```

Und nun:

```
PRINT FRE (0)-(SGN(FRE(0))<0)*65535
```

Sehen Sie die Änderung? Das Basic nimmt nun an, es stehe weniger Speicherkapazität zur Verfügung. In diesem gewonnenen Speicherplatz können Sie nun Ihren Zeichensatz unterbringen. Als nächstes sind die Zeichen im RAM zu definieren. Zu Beginn stehen ab 12288 ($3000) zufällige Daten. Durch folgendes Programm werden 64 Zeichen vom ROM- in

WERT VON A	BITS	BANK	START-PLATZ	BEREICH DES VIDEOCONTROLLER BAUSTEINS
0	00	3	49152	($C000-$FFFF)
1	01	2	32768	($8000-$BFFF)
2	10	1	16384	($$000-$7FFF)
3	11	0	0	($0000-$3FFF) (Standardwert)

2. Alle möglichen 16-KByte-Bänke des Videocontrollers (56578 / 56576)

Wert	Bits	dezimal	hexadezimal
0	XXXX000X	0	$0000-$07FF
2	XXXX001X	2048	$0800-$0FFF
4	XXXX010X	4096	$1000-$17FF ROM-Image in Bank 0 & 2 (Standard)
6	XXXX011X	6144	$1800-$1FFF ROM-Image in Bank 0 & 2
8	XXXX100X	8192	$2000-$27FF
10	XXXX101X	10240	$2800-$2FFF
12	XXXX110X	12288	$3000-$37FF
14	XXXX111X	14336	$3800-$3FFF

3. Die Bits 1, 2, und 3 des Kontrollregisters 53272 bestimmen die Lage des Zeichensatzspeichers

den RAM-Zeichensatz übertragen:

```
5 PRINT CHR$(142)
10 POKE 52,48:POKE 56,48:CLR
20 POKE 56334,PEEK (56334) AND 254
30 POKE 1,PEEK (1) AND 251
40 FOR I = 0 TO 511:POKE I +12288,PEEK(I+53248):NE-
XT
50 POKE 1,PEEK(1)OR4
60 POKE 56334,PEEK(56334)OR1
70 END
```

Geben Sie nun ein:

```
POKE 53272,(PEEK(53272)AND240)+12
```

Nichts passiert, stimmt's? Fast nichts! Der C 64 bekommt die Zeicheninformationen nun vom RAM und nicht vom ROM. Da wir jedoch die Zeichen genau aus dem ROM kopiert haben, ist kein Unterschied zu sehen ...noch nicht. Die Zeichen können nun leicht geändert werden. Löschen Sie den Bildschirm und drücken Sie <@>. Bewegen Sie den Cursor um einige Zeilen nach unten, und geben Sie dann ein:

```
FOR I = 12288 TO 12288 + 7:POKE I,255-PEEK(I):NEXT
```

Sie haben soeben ein <@> in Revers geschaffen.

Bewegen Sie nun den Cursor wieder zum Programmanfang und tippen Sie <RETURN> erneut, um das Zeichen noch einmal umzukehren. Die Tabelle der Bildschirmcodes zeigt Ihnen, wo die einzelnen Zeichen im RAM stehen. Denken Sie daran, daß zur Speicherung jedes Zeichens acht Speicherplätze benötigt werden. Wird ein anderes Zeichen gewünscht, ist vorher noch etwas zu berücksichtigen. Was ist zu tun, wenn Sie nun Zeichennummer 154, ein umgekehrtes Z, wünschen? Sie können das erreichen, indem Sie ein Z umkehren, oder Sie können den Satz der umgekehrten Zeichen aus dem ROM kopieren oder einfach das eine Zeichen aus dem ROM holen und ein nicht benötigtes Zeichen im RAM dadurch ersetzen. Nehmen wir an, Sie benötigen das Zeichen > nicht mehr. Es soll also gegen das negative dargestellte Zeichen Z ausgetauscht werden. Geben Sie ein:

```
FOR I = OTO 7:POKE 12784+I,255-PEEK(I+12469):NEXT
```

(Diese Änderung betrifft jedoch nur die Darstellung auf dem Bildschirm. Auch wenn das Zeichen wie ein umgekehrtes Z aussieht, wirkt es in einem Programm doch immer noch

als >):Probieren Sie das an einem Beispiel aus, bei dem dieses Zeichen benötigt wird.

Kommen wir nun zur Kreation eigener Zeichen. Wissen Sie noch, wie Zeichen im ROM gespeichert sind? Die Bitmuster der Zeichenbytes geben direkt das Zeichen wieder. Werden 8 Byte übereinander angeordnet und jedes Byte als achtstellige Binärzahl geschrieben, entsteht eine 8 x 8-Matrix, die das Zeichen darstellt. Enthält ein Bit eine 1, ist an diesem Platz ein Punkt; enthält es eine 0, ist an diesem Platz eine Leerstelle.

Multicolor-Zeichen

Durch die standardmäßige hochauflösende Grafik können Sie selbst Einzelpunkte auf dem Bildschirm ansteuern. Für jeden Punkt im Zeichensatz stehen zwei Werte zur Verfügung: 1 für EIN und 0 für AUS. Hat ein Punkt den Wert 1, wird er in der von Ihnen für die jeweilige Bildschirmposition gewählten Farbe angezeigt. Bei der hochauflösenden Grafik können alle Punkte innerhalb der 8 x 8-Matrix entweder in der Hinter- oder Vordergrundfarbe angezeigt werden. Hierdurch wird die Farbauflösung innerhalb dieses Bereichs eingeschränkt. So können z.B. Schwierigkeiten entstehen, wenn sich zwei Linien mit verschiedenen Farben kreuzen.

Dieses Problem wird durch den Mehrfarbenmodus gelöst. Hierbei kann jeder Punkt eine von vier Farben annehmen: Bildschirmfarbe (Hintergrundfarbregister #0), die Farbe im Hintergrund #1, die Farbe im Hintergrundfenster #2 oder die Zeichenfarbe. Die einzige Einschränkung liegt in der horizontalen Auflösung, da im Mehrfarbenmodus jeder Punkt doppelt so breit ist wie bei Hochauflösung. Es überwiegen jedoch bei weitem die vielen Vorteile des Mehrfarbenmodus.

Zum Einschalten des Modus für mehrfarbige Zeichen wird Bit 4 des Steuerregisters durch folgende POKE-Anweisung bei 53270 ($ D 016) gesetzt.:

```
POKE 53270,PEEK(53270)OR 16
```

Zum Abschalten dieser Betriebsart wird Bit 4 der gleichen Speicherzelle (53270) durch nachstehende POKE-Anweisung auf 0 gesetzt:

```
POKE 53270,PEEK(53270)AND239
```

Dieser Mehrfarbenmodus wird für jede Bildschirmstelle ein- oder ausgeschaltet, so daß Mehrfarbengrafiken und Grafiken mit hoher Auflösung (HiRes) kombiniert werden können. Dies wird über Bit 3 im Farbspeicher gesteuert. Der Farbspeicher beginnt ab 55296 ($ D800). Ist die Zahl im Farbspeicher kleiner als 8 (0-7), so gilt für die entsprechende Stelle auf dem Bildschirm Hochauflösung in der gewählten Farbe (0-7). Ist die Zahl im Farbspeicher größer oder gleich 8 (8-15), wird die entsprechende Stelle im Mehrfarbenmodus angezeigt.

Die Zeichenfarbe einer Bildschirmposition kann durch eine POKE-Anweisung im Farbspeicher geändert werden. Durch das POKEn einer Zahl von 0 bis 7 werden die Zeichen in normaler Farbdarstellung angezeigt. Durch das POKEn einer Zahl zwischen 8 und 15 gilt für die entsprechende Bildschirmstelle der Mehrfarbenmodus. Das heißt durch Einschalten von Bit 3 im Farbspeicher wird der Mehrfarbenmodus und durch Ausschalten der Hochauflösungsmodus gewählt.

Gilt für eine Bildschirmstelle der Mehrfarbenbetrieb, wird durch die Zeichenbits bestimmt, welche Farben für die Punkte angezeigt werden. Sie sehen die Darstellung des Buchstabens A und das entsprechende Bit-Muster in Abb. 2.

Im normalen oder HiRes-Modus wird die Bildschirmfarbe bei jedem 0-Bit und die Zeichenfarbe stets da gezeigt, wo das Bit 1 ist. Beim Mehrfarbenmodus werden die Bits paarweise benutzt (Abb. 3).

Im Bildbereich werden die durch AA gekennzeichneten Stellen in der Hintergrundfarbe #1, die durch BB gekennzeichneten Stellen in der Hintergrundfarbe #2, und die durch CC gekennzeichneten Stellen in der Zeichenfarbe dargestellt. Dies wird durch Bitpaare bestimmt (Tabelle 4).

Extended-Color-Modus

In diesem Modus können Sie für jedes einzelne Zeichen sowohl die Hintergrund- als auch die Vordergrundfarbe steuern. So ist es z.B. möglich, auf einem weißen Bildschirm ein blaues Zeichen mit gelbem Hintergrund zu zeigen. Für den erweiterten Hintergrundfarbmodus stehen vier Register parat. Für jedes kann eine der 16 Farben gewählt werden.

In diesem Modus wird über den Farbspeicher die Vordergrundfarbe festgelegt. Die Anwendung ist die gleiche wie beim Standardzeichenmodus.

Bei erweitertem Modus ist die Anzahl der verschiedenen anzeigbaren Zeichen jedoch eingeschränkt. Ist der erweiterte Farbmodus eingeschaltet, werden nur die ersten 64 Zeichen des Zeichen-ROM (oder die ersten 64 in Ihrem programmierbaren Zeichensatz) verwendet. Zwei Bit des Zeichencodes werden nämlich für die Wahl der Hintergrundfarbe benutzt.

Der Zeichencode (die auf dem Bildschirm gePOKEte Zahl) vom Buchstaben A ist eine 1. Im erweiterten Farbmodus erscheint nach dem POKEn einer 1 ein A. Normalerweise muß nach dem POKEn von 65 das Zeichen mit dem Zeichencode (CHR$) 129, also ein invertiertes A, erscheinen. Dies passiert nicht im erweiterten Farbmodus. Es erscheint genau dasselbe A wie vorher, jedoch eine andere Hintergrundfarbe. Entnehmen Sie die Codes der Tabelle 5.

Zum Einschalten des erweiterten Farbmodus wird Bit 6 des VIC-II-Registers mit der Adresse 53265 ($D001) auf 1 gesetzt. Dies geschieht durch:

```
POKE 53265,PEEK(53265)OR 64
```

Zum Ausschalten des erweiterten Farbmodus wird Bit 6 des VIC-II-Registers mit der Adresse 53265 ($ D 011) auf 0 gesetzt. Hierzu führt:

```
POKE 53265,PEEK(53265)AND 191
```

Betrachten Sie dazu auch:

```
LOAD"EXTENDED.COLOR",8
```

Es läßt sich mit RUN starten.

Zum Betrachten der einzelnen Modi befindet sich ein BASIC-Programm auf der beiliegenden Diskette:

```
LOAD"BETRACHTER",8
```

lädt dieses und RUN startet es. Geben Sie im ersten Schritt den Namen des zu ladenden Zeichensatzes an. Auf der Diskette befinden sich 14 davon (»20 bis 50«). »Betrachter kann alle Modi simulieren (Abb. 4). Zum Schluß noch ein Hinweis:

Sollten Sie eine Erweiterung eingeschaltet haben, oder tritt beim Laden die Meldung »FEHLER« auf, obwohl Sie den Zeichensatz richtig eingetippt haben, löschen Sie einfach die Programmzeile 270. (gr)

BIT-PAAR	FARBREGISTER	SPEICHERPLATZ
00	HINTERFGRUNDFARBE # 0 (BILDSCHIRMFARBE)	53281 ($D021)
01	HINTERGRUNDFARBE # 1	53282 ($D022)
10	HINTERGRUNDFARBE # 2	53283 ($D023)
11	DURCH DIE UNTEREN 3 BITS IM FARBSPEICHER BESTIMMTE FARBE	FARBSPEICHER

4. Mögliche Farbkombinationen für Multicolorzeichen

ZEICHENCODE			HINTERGRUNDFARBREGISTER	
BEREICH	BIT 7	BIT 6	NUMMER	ADRESSE
0 - 63	0	0	0	53281($D021)
64 - 127	0	1	1	53282($D022)
128 - 191	1	0	2	53283($D023)
192 - 255	1	1	3	53284($D024)

5. Im Hintergrundfarbmodus können die ersten 64 Zeichen mit unterschiedlichen Hintergrundfarben dargestellt werden

C64 im Hires-Modus

1 PREMIERE FÜR HOHE AUFLÖSUNG

Grafik – einer der Glanzpunkte des C64! Nur: mit Basic 2.0 sehr umständlich zu realisieren. Besser geht's mit Assembler.

Bereits beim Einschalten des C64 befindet er sich in einem Grafikmodus: in der niedrigen Auflösung (engl. Low Resolution, Abk. Lores). Es steht ein Bildschirmspeicher von 1000 Byte zur Verfügung: 25 Zeilen mit jeweils 40 Spalten. Dieser Bereich umfaßt die Adressen $0400 (1024) bis $07E7 (2023). Jedes Zeichen, das über die Tastatur eingeben wird (Buchstaben, Zahlen, Blockgrafik), besitzt eine charakteristische Bildschirm-Codezahl. Diese merkt sich der C64, multipliziert sie intern mit »8« und sucht im Charakter-ROM (Zeichensatzspeicher ab $D000, dez. 53248) die Adresse, die sich nach folgender Formel ergibt:

```
Zeichenadresse = Zeichensatzanfang + 8 * Codewert
```

Nun werden noch die dahinterliegenden sieben Bytes dazugenommen und das gesamte Zeichenmuster auf den Bildschirm gebracht. Noch ausführlicher steht's in unserem Grundlagenartikel zum Zeichensatz in diesem Sonderheft.

Uns interessiert im Zusammenhang mit dem Zeichensatz der Aufbau eines Zeichenmusters: Acht Byte, die jeweils aus acht Bit bestehen. Ein Zeichen besitzt also eine Matrix von 8 x 8 = 64 Bildpunkten. Das gilt z.B. auch für ein Leerzeichen (<SPACE>), bei dem gar nichts zu sehen ist: Es sind trotzdem 64 Pixel, nur besitzt jeder Bildpunkt eben den Wert »0«.

Ließen sich im Lores-Modus (oder Textbildschirm) nur jeweils Päckchen von 8 x 8 Bit manipulieren (anzeigen, löschen) kann der hochauflösende Grafikmodus (engl. High Resolution, Abk. Hires) bedeutend mehr – jedes einzelne Bit der 1000 möglichen Bildschirmzeichen kann beliebig ein- oder ausgeschaltet werden. Der Textbildschirm wird zur Grafiklandkarte. So eine Bitmap umfaßt jetzt 1000 x 8 x 8 = 64000 Bildpunkte (= Bits) oder umgerechnet: 8000 Byte.

Wie läßt sich dieser Zustand beim C64 aktivieren? Dazu gibt's drei wichtige Adressen:

Speicherstelle $D011 (53265)

Das ist Register 17 des Video Interface Controllers (VIC). Jedes Bit des Werts, der in dieser Adresse steht, fungiert als Schalter: 1 = Funktion an, 0 = aus. Die einzelnen Schalter bedeuten:
- Bits 0 bis 2: Soft Scrolling,
- Bit 3: 25 oder 24 Zeilen im Textmodus,
- Bit 4: Bildschirm ein/aus
- Bit 5: Hires-/Lores-Modus
- Bit 6: Extended Color Modus ein/aus
- Bit 7: 9. Bit des Speicherinhalts der Adresse $D012 (dieses Register ist nur bei der Programmierung des Rasterzeilen-Interrupts von Bedeutung).

Der Einschaltwert von Adresse $D011 ist »27«. In einzelne Bits zerlegt (binär), sieht das so aus:

```
Bit 7 6 5 4 3 2 1 0
 -------------------
    0 0 0 1 1 0 1 1
```

Da wir uns noch im Textmodus befinden, steht in Bit 5 eine Null. Um es einzuschalten, müssen wir die Wertigkeit »32« zum Normalinhalt der Adresse $D011 addieren:

```
5 SYS 58692: REM LORES-BILDSCHIRM LOESCHEN
10 POKE 53265,PEEK(53265)+32
```

Nach RUN ist der Hires-Modus aktiv, nur: Auf dem Bildschirm erscheint ein wirres Byte-Chaos (sogar Muster aus dem Zeichensatz sind zu erkennen). Mit <RUN/STOP RESTORE> verbannen Sie den häßlichen Screen: Was Sie gesehen haben, waren die ersten 8000 Speicherstellen des C64 im Einzelbit-Modus: Zeropage, Stapelspeicher, Sprungvektoren, Bildschirm- und Basic-Bereich bis zur Adresse 7999.

Um den Hires-Bildschirm zu löschen, müßte man per Schleife jedes Byte auf »0« setzen:

```
FOR I=0 TO 7999: POKE I,0: NEXT
```

aber: Geben Sie's besser noch nicht ein – denn sonst löschen Sie automatisch Bytewerte (z.B. in der Zeropage), die für den C64 lebenswichtig sind. Wir müssen uns also einen Platz suchen, in dem wir den 8000 Byte großen Hires-Bereich unterbringen können und der die Schaltkreise des Computers nicht stört. Er muß allerdings innerhalb der ersten 16384 Adressen des C64 liegen – mehr Bytes kann der VIC-Chip nicht überblicken. Man nennt diesen Bereich VIC-Bank 0 (alle folgenden Routinen gelten nur für diese Bank). Insgesamt gibt's vier davon – geregelt wird das in Adresse $DD00 (56576). Wer mehr darüber wissen möchte, dem empfehlen wir unseren großen Grafikkurs im 64'er-Sonderheft 45.

Der ideale Speicherplatz einer Bitmap in VIC-Bank 0 liegt ab Speicherstelle $2000 (8192). Um dies dem C64 mitzuteilen, brauchen wir Register 24 des VIC-Chip:

Speicherzelle $D018 (53272)

Sie dient ebenfalls als Schalter für diverse Aufgaben:
- Bit 0: nicht benutzt, ist immer eingeschaltet (1),
- Bit 1 bis 3: Startadresse des Charakter-ROM,
- Bit 4 bis 7: Startadresse Bildschirm-RAM.

Die Bitbelegung des Einschaltwerts »21« gibt uns die Info:

```
Bit 7 6 5 4 3 2 1 0
-------------------
    0 0 0 1 0 1 0 1
```

Wir müssen jetzt das Byte in zwei Hälften gespaltet betrachten: Die Bits 0 bis 3 sind das Low-Nibble, Nr. 4 bis 7 bezeichnet man als High-Nibble. Wenn man Bit 0 (immer »1«) außer acht läßt, steht im Low-Nibble die Zahl »4«, im High-Nibble der Wert »1«. Multiplizieren Sie beide Zahlen mit »1024« – es stimmt: Der Zeichensatz beginnt bei »4096«. Bevor Sie reklamieren: Es ist richtig, daß das Charakter-ROM bei Adresse $D000 (53248) beginnt, aber es wird intern nach $1000 (4096) gespiegelt – sonst läge es nicht innerhalb der ominösen VIC-Bank von 16384 Byte.

Als Adresse fürs Bildschirm-RAM bekommen wir »1024«. Warum ist der Bildschirmspeicher in der hochauflösenden Grafik so wichtig, wenn er doch nur im Textmodus Zeichen auf den Monitor bringt? Er wird zum Farb-RAM umfunktioniert: Bei der Hires-Grafik stehen die bekannten 16 verschiedene Werte für Vorder- und Hintergrundfarbe zur Verfügung, die gespeichert werden müssen.

Zurück zu unserer geplanten Anfangsadresse für die Bitmap: »8192«, geteilt durch »1024«, ergibt »8«. Im Low-Nibble

steht bereits die Zahl »4«, also muß der Bytewert von $D018 nur noch um »4« erhöht werden:

```
20 POKE 53272,PEEK(53272)+4
```

Ab dieser Adresse läßt sich die Bitmap löschen, ohne Schaden anzurichten:

```
30 FOR I=8192 TO 8192 + 7999
40 POKE I,0: NEXT
```

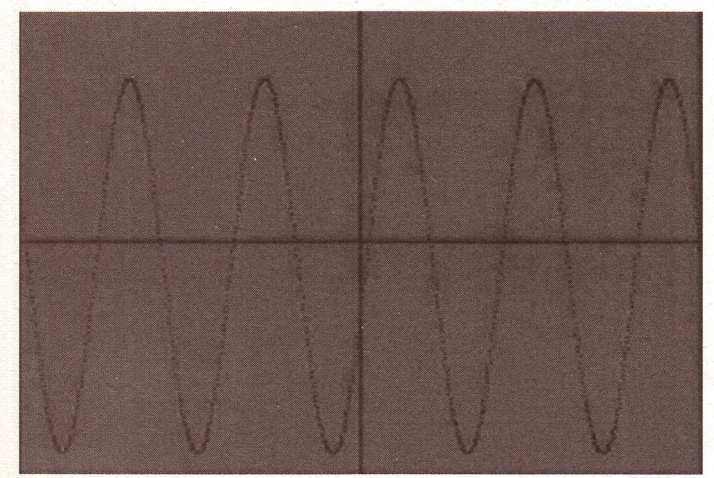
Hängen Sie noch folgende Zeilen dran:

```
50 POKE 198,0: WAIT 198,1
60 POKE 53265,27: POKE 53272,21
```

Nach Druck auf eine beliebige Taste ist wieder alles wie gehabt.

Möchten Sie's mit einer Grafik von der Sonderheft-Diskette ausprobieren? Dann ergänzen Sie das Basic-Listing um folgende Zeile:

```
1 IF A=0 THEN A=1: LOAD "DEMO1.PIC",8,1
```



[1] Eine Sinus-Kurve, entstanden mit der Assembler-Routine PLOT

```
DY GRAPHIDDY GRAPH

Clear Bitmap
Save Picture
Load Picture
Draw Lines    ←
End/Reset
<Menu : <M>>
```

[2] Die wichtigsten Arbeitsfunktionen ruft man mit <RETURN> auf

Legen Sie die Diskettenseite mit dem Hires-Bild »Demo1.Pic« (im Programmpaket zu »Hi-Eddi«) ins Laufwerk und starten mit RUN.

Wer jetzt Grafikpunkte (=Pixel) auf dem Hires-Bildschirm einschalten möchte, steht vor einem neuen Problem: Wie teilt man dem Computer die entsprechende Position des Bildpunkts im Koordinaten-System 320 x 200 mit?

Für den C64 existiert theoretisch noch immer die Bildschirmeinteilung des Lores-Modus: 25 Zeilen mit je 40 Spalten. Als Basisadresse für den Beginn des Hires-Bereichs legt man die Speicherzelle »8192« fest: BA = 8192. Das Farb-RAM solcher Hires-Grafiken liegt immer bei $0400 (1024).

Ein Beispiel: Wir wollen das Pixel an folgenden Koordinatenpunkten bestimmen: Horizontal (x-Richtung): 160, vertikal (y-Richtung): 100.

```
ZEILE = INT(100/8)
(Ergebnis: 12)
```

Die zugehörige Spalte berechnet man mit dieser Formel:

```
SPALTE = INT(160/8)
(Ergebnis: 20)
```

Damit steht die Position der 8 x 8-Pixel-Matrix des Zeichens auf dem Bildschirm fest, die den gewünschten Bildpunkt enthält: Zeile 12, Spalte 20.

Jetzt kommt die Feineinstellung: Das Byte an dieser Position (LINIE) besteht aus acht Pixeln:

```
LINIE = 100 AND 7
(Ergebnis: 4)
```

Zum entsprechenden Bit bzw. Bildpunkt kommt man mit dieser Formel:

```
BIT = 7 - (160 AND 7)
(Ergebnis: 7)
```

Per POKE ins betreffende Byte der Hires-Grafik wird dann exakt dieses Pixel eingeschaltet. Dazu muß man die bekannten Daten verwenden: Basisadresse = 8192, maximale x-Richtung (320), Position des Zeichenmusters x 8 und die Ergebnisse der Berechnungen von LINIE und BIT:

```
BYTE = BASIS + ZEILE*320 + SPALTE*8 + LINIE
(Ergebnis: BYTE = 8192+12*320+20*8 + 4 = 12196)
```

Mit folgender POKE-Anweisung erscheint der Grafikpunkt:

```
POKE 12196,PEEK(12196) OR 2↑7
```

Eines ist klar: Hires-Grafik mit Basic 2.0-Befehlen zu erzeugen, ist sehr zeitraubend. Im 64'er-Sonderheft 69 (Basic) ist das mit vielen Beispielprogrammen ausführlich beschrieben. Wir widmen uns lieber Assembler-Routinen, die per SYS-Befehl in Basic aufgerufen werden und alle Funktionen blitzschnell erledigen: Bitmap und Bildschirmspeicher löschen, Farben einstellen, Grafikpixel setzen usw.

Dazu finden Sie jede Menge Assembler-Routinen auf der Sonderheft-Diskette. Alle Programme müssen mit der Endung »,8,1« geladen und per SYS-Befehl gestartet werden:

- GRAPHICCLR (SYS 28700): löscht die Bitmap im Bereich von 8192 bis 16383. Speicherbelegung: $701C bis $7035.
- BITMAPON (SYS 28726): schaltet den hochauflösenden Grafikmodus in den VIC-Registern 17 und 22 ein. Speicherbelegung: $7036 bis $7046.
- BITMAPOFF (SYS 28743): Der Hires-Modus wird wieder abgeschaltet. Jetzt ist der Textbildschirm wieder aktiv. Speicherbereich: $7047 bis $7056.
- COLORS (SYS 28759): Mit diesem Utility lassen sich die Bildschirmfarben per Tastendruck ändern:

Farbtasten <1> bis <8> mit <CTRL>: Vordergrundfarbe, Farbtasten <1> bis <8> mit <SHIFT>: Hintergrundfarbe. Probieren Sie's aus:

```
10 SYS 28700: SYS 28726
20 SYS 28759: GOTO 20
```

Voraussetzung: Sie müssen die Dateien GRAPHICCLR, BITMAPON und COLORS vorher laden. Das Farbänderungsprogramm belegt den Speicher von $7057 bis $70DB.

- PLOT (SYS 28892): Das ist die Assembler-Routine zum Setzen eines Grafikpunkts auf dem Bildschirm. Möchten Sie mehrere Pixel hintereinander einschalten, muß dies per Basic-Schleife geschehen (FOR-NEXT). Speicherbereich: $70DC bis $71B5.

Zwei Demoprogramme auf unserer Diskette zeigen, wie man die PLOT-Funktion in eigenen Basic-Programmen verwendet: SPRITERASTER erzeugt eine 24 x 21 große Quadrat-Matrix auf dem Hires-Bildschirm, die exakt der Bit-Belegung eines Sprite-Musters entspricht. Wenn Sie zuvor HARDCOPY (s. Beschreibung) geladen haben und nach dem Programmende »SYS 33710« im Direktmodus eingeben,

können Sie das Sprite-Raster beliebig oft ausdrucken und für eigene Sprite-Entwürfe verwenden.

SINUS zeichnet eine Kurve nach vorgegebenen Funktionen (Abb. 1). Nach dem Zeichnen der Grafik läßt sie sich ebenfalls speichern oder ausdrucken.

- SAVEPIC (SYS 31290): speichert eine Hires-Bitmap auf Diskette (33 Blöcke). Vorher werden Sie nach dem Filenamen gefragt, der nicht länger als zwölf Zeichen sein darf - die Endung ».PIC« wird automatisch angefügt. Speicherbelegung: $7A3A bis $7B10.

- LOADPIC (SYS 31098): lädt ein beliebiges Grafikbild im Hi-Eddi-Format (Hires-Bereich ab $2000). Speicherbereich: $797A bis $7A39.

Sie können es sofort ausprobieren, wenn Sie dazu das Beispielprogramm »Bildlader« in den Computer holen:

```
LOAD "BILDLADER",8
```

Nach dem Start mit RUN werden Sie nach dem Namen der Grafik und den gewünschten Farben gefragt, die Sie beliebig einstellen können. Jetzt löscht das Programm per SYS-Befehl den Hires-Bildschirm, schaltet die hochauflösende Grafik ein und lädt das Bild. Anschließend erscheint die Hires-Grafik auf dem Schirm. Ein Tastendruck bringt Sie zurück in den Lores-Modus.

- HARDCOPY (SYS 33710): Ein Grafikbild wird auf jedem Commodore-Drucker oder Epson-Gerät mit kompatiblem In-

Zeichenprogramm ohne Joystick

terface ausgegeben. Dazu muß der Hires-Modus nicht aktiv sein. Wenn »Hardcopy« im Speicher steht, muß man nur das gewünschte Grafikbild laden (mit »,8,1«) und die Routine mit SYS 33710 starten. Nach jedem Ausdruck setzt das Programm automatisch einen Zeilenvorschub. Da alle Bilder im Hi-Eddi-Format gespeichert werden, lassen sich selbstgenerierte Grafiken auch mit »Hi-Print« aus dem Programmpaket »Hi-Eddi« ausdrucken oder mit jedem anderen Hardcopy-Programm zur Hires-Grafik (s. 64'er-Sonderheft 72, Thema »Drucker«). Speicherbelegung des Utilities: $83A7 bis $848A. Alle bisher genannten Assembler-Routinen fügen sich nahtlos aneinander. Ein Kinderspiel, sie zu einem Gesamtprogramm zusammenzufassen. Zur Funktionssteuerung muß man sie nur ergänzen und mit einem Auswahlmenü versehen. Im folgenden Programm wurde dies verwirklicht:

```
LOAD "GRAPHIDDY",8
```

Nach dem Start mit RUN werden die inzwischen bekannten Teilroutinen an die vorgesehenen Speicherbereiche transponiert (und noch einige mehr: Steuerung des Bleistift-Sprites, Tastaturabfrage, Text in Grafik einbinden, Projektionen, Menü und Füllen von Flächen) Das Arbeitsmenü erscheint auf dem Bildschirm (Abb. 2). Hinter den einzelnen Menüpunkten steht der <Pfeil links>, den Sie mit den Cursortasten auf- oder abwärts bewegen können. Die entsprechende Funktion wird mit <RETURN> initialisiert:

Clear Bitmap: ruft die Routine GRAPHICCLR auf. Der Hires-Bildschirm wird vom Bytemüll befreit. Diese Operation geschieht verdeckt - ohne Einschalten des Hires-Bildschirms.

Save Picture: speichert eine Hires-Grafik nach Eingabe des entsprechenden Filenamens auf Diskette. Achtung: Der Name darf maximal zwölf Zeichen lang sein, da die Endung ».PIC« automatisch angehängt wird. Zahlen sind nicht erlaubt, nur Buchstaben.

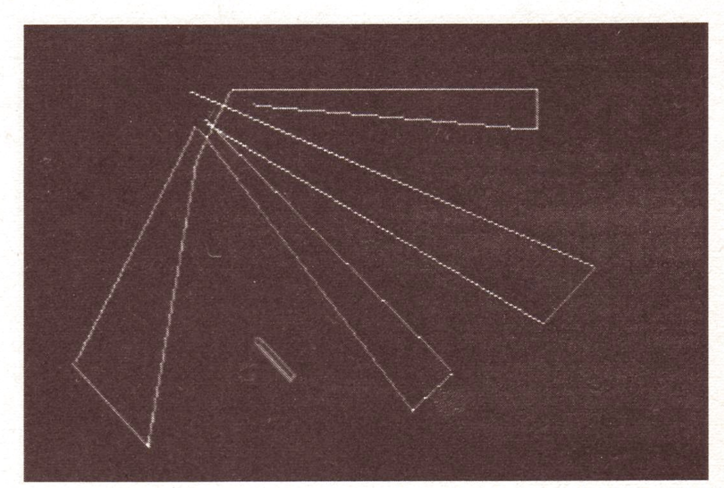
Load Picture: lädt eine Hires-Grafik von Diskette mit der Endung »PIC«, die im Hi-Eddi-Format gespeichert wurde. Fremde Grafiken müssen notfalls umbenannt werden. Die Endung »PIC« hängt das Programm automatisch an den Filenamen, nach dem Sie gefragt werden. Es können nur Schwarzweiß-Grafiken geladen werden (ohne Farb-RAM).

Draw Lines: schaltet die Hires-Grafik ein. Die Zeichenfläche erscheint. Mit dieser Funktion stehen Ihnen alle Zeichenfunktionen von »Graphiddy« zur Verfügung. Sie lassen sich aktivieren durch:

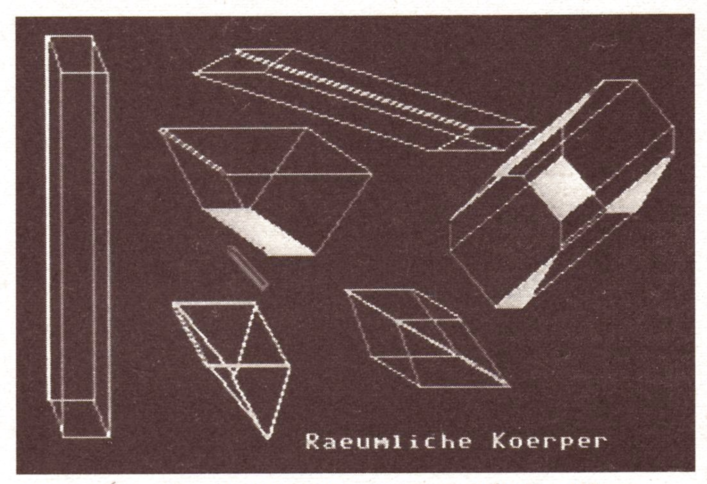
<**B**>: Freihand-Zeichnen. Nach anschließendem Druck auf <F> wird der Modus »Punkte setzen« aktiviert: Der Bleistift-Sprite folgt allen vier Richtungen der Cursortasten und hinterläßt Grafiklinien (Abb. 3). Weitere Richtungen stehen mit der Schrägstrich- und der Punkttaste zur Verfügung:

- </>: schräg rechts abwärts,
- <SHIFT />: schräg links aufwärts,
- <.>: schräg links abwärts,
- <SHIFT .>: schräg rechts aufwärts.

Erneuter Druck auf <F> stellt das wieder ab: Dann kann der Sprite beliebig positioniert werden, ohne Grafikpunkte zu setzen.



[3] Freihand-Zeichnen ohne Joystick: Der Sprite-Bleistift wird ausschließlich per Tastatur gezeichnet



[4] Dreidimensionale Projektionen geometrischer Körper

<**R**>: Diesen Modus muß man einschalten, um gesetzte Linien zu löschen. Die Funktion arbeitet pixelgenau: Sie müssen die gezeichneten Linien exakt überfahren, um sie zu löschen. Mit <Z> schaltet man den Freihand-Zeichnen-Modus wieder ein.

<**G**>: zieht eine Linie, die sich mit dem Cursor bewegt. Sie läßt sich beliebig ausdehnen. Die Cursorgeschwindigkeit ist durch diese Funktion natürlich langsamer. Durch erneuten Druck auf <B> läßt sich diese Funktion abschalten.

<**D**>: Damit können Linien gezogen werden. Vorher muß man den Zeichenbleistift in die gewünschte Position bringen. Dann drückt man <P>, um den Linienanfang zu bestim-

men. Bewegen Sie jetzt den Sprite auf einen beliebigen Endpunkt. Erneuter Druck auf <P> zeichnet die Linie.

<**T**>: aktiviert die Texteingabe im Grafikbildschirm an der aktuellen Position des Bleistift-Sprites, der dann unsichtbar wird. Es sind alle Zeichen der Tastatur erlaubt; Fehleingaben lassen sich mit <DEL> löschen. Mit <RETURN> verläßt man diesen Modus – das Sprite erscheint wieder.

Das ist noch nicht alles: Im Zeichenmodus lassen sich auch 3-D-Projektionen entwerfen. Entwerfen Sie die Vorderansicht eines Körpers auf den Bildschirm (Linien mit <P> verbinden). Anschließend drückt man <V>, um den Projektionsmodus einzuschalten. Jetzt läßt sich der Bleistift-Sprite an jede gewünschte Position auf dem Bildschirm setzen. Per Taste <A> wird die Projektion ausgeführt: Auf dem Monitor entsteht ein dreidimensionales Gebilde (Abb. 4). Mit <Q> kann man diesen Modus verlassen.

<**E**>: Bewegen Sie das Sprite auf die Fläche, die sich mit der aktuell eingestellten Farbe füllen soll. Nach Druck auf <E> färbt sich die Fläche. Werden bestimmte Teile nicht berücksichtigt, müssen Sie den Bleistift-Sprite dahin bewegen und erneut <E> drücken.

<**F3**>: vergrößert den Bleistift-Sprite in alle vier Richtungen. Jeder Druck auf die Cursortasten rückt ihn nur um ein Pixel weiter. Diese Funktion dient der Feinabstimmung beim Entwurf einer Hires-Grafik.

<**F5**>: normalisiert Größe und Geschwindigkeit des Sprite-Bleistifts. Die Cursortasten schieben ihn jetzt wieder acht Pixel nach vorn. Damit kann man größere Entfernungen zwischen zwei Grafikkoordinaten schnell überwinden.

Kurzinfo: Graphiddy

Programmart: Zeichenprogramm
Laden: LOAD "GRAPHIDDY",8
Starten: nach dem Laden RUN eingeben
Steuerung: Tastatur
Besonderheiten: Mit den Tasten <D>, <V> und <A> lassen sich 3-D-Projektionen erzeugen
Benötigte Blocks: 23

<**F7**>: gibt eine Hires-Grafik auf Epson-kompatiblen Druckern aus, die per Hardware-Interface mit dem seriellen Port des C64 verbunden sind.

<**S**>: ruft im aktiven Grafikmodus die Funktion »Grafik speichern« auf.

<**L**>: lädt eine Hires-Grafik im Hi-Eddi-Format (Speicheranfang: $2000).

<**M**>: bringt Sie zurück ins Menü von »Graphiddy«.

Im Programm ist auch die Routine COLORS integriert. Die Hintergrundfarben lassen sich in jedem aktiven Grafikmodus mit den Tastenkombinationen aus <SHIFT> und <1> (schwarz) bis <8> (gelb) ändern. Die Zeichen- bzw. Vordergrundfarbe kann man mit denselben Zahlentasten, aber dann gemeinsam mit <CTRL> umstellen.

Hinweise zum Programm

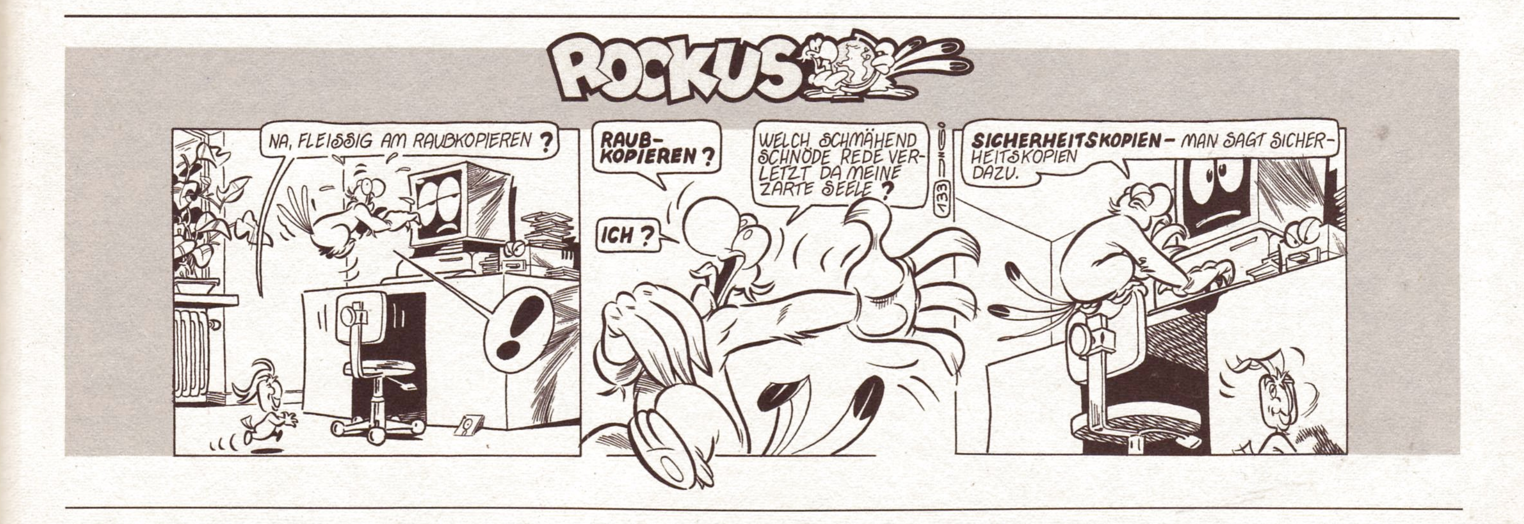
Nach dem Laden belegt »Graphiddy« den Basic-Speicher von $0801 (2049) bis $1E90 (7824). Durch den Start mit RUN werden die Daten in den vorgesehenen Bereich von $7000 (28672) bis $8650 (34384) verschoben und der Basic-Anfang nach $4000 (16384) verlegt. Somit bleiben 12 288 Bytes für eigene Programmentwicklungen frei, in denen man per SYS-Befehl die genannten Grafik-Routinen einzeln schalten kann. Das entspricht 48 Blocks auf Diskette. Achten Sie darauf, daß Ihr eigenes Basic-Programm nicht größer wird: Es überschreibt sonst die Routinen von »Graphiddy«.

Der Bereich des Kassettenpuffers wird ausgiebig zur Ablage von temporären Werten benutzt (Flags, Koordinaten für x- und y-Richtung, zergliedert in High- und Low-Byte). Aus diesem Grund kann man »Graphiddy« (oder Einzelroutinen daraus) nur im Zusammenhang mit einer Floppy-Station benutzen (1541 oder 1571), nicht mit der Datasette.

»Graphiddy« lädt auch fremde Hires-Grafiken. Voraussetzung: Sie müssen im Hi-Eddi-Format vorliegen (Speicherbereich: $2000 (8192) bis maximal $3FFF (16383)) und einfarbig sein. Außerdem dürfen diese Bilder kein Farb-RAM besitzen. Man erkennt solche Dateien am Umfang des Speicherplatzes, den sie auf Diskette belegen: 37 Blocks. Die Grafiken kann man mit den Tastaturbefehlen des Programms nachbearbeiten (verändern, ergänzen) und erneut im kompatiblen Grafikformat unter einem anderen Dateinamen speichern. Selbstverständlich ist auch der umgekehrte Weg möglich: Sämtliche Hires-Bilder aus »Graphiddy« lassen sich mit jedem anderen C-64-Zeichenprogramm laden, nachbearbeiten und speichern (z.B. Hi-Eddi, Starpainter, OCP-Art-Studio usw.) Geänderte Grafiken lassen sich auch jederzeit wieder mit den genannten Zeichenprogrammen laden.

Mit der eingebauten 3-D-Funktion von »Graphiddy« kann man – zumindest im groben Rahmen – sogar einfache, technische Zeichnungen auf dem Bildschirm realisieren. Sie werden rasch selbst herausfinden, welchen Nutzen unser Zeichenprogramm ohne Joystick für Sie haben kann.

Wem es zu umständlich ist, Grafiken mit dem Joystick zu zeichnen, findet in »Graphiddy« eine komfortable Alternative. Außerdem lassen sich sämtliche Einzelroutinen in eigenen Basic-Programmen verwenden. (bl)



Zwischen damals, als man C-64-Grafik noch mit PRINT-Befehlen und geSHIFTeten Tastaturzeichen auf den Schirm brachte, und heute, da ausgeschlafene Programmierer aus dem C 64 eine fast professionelle Grafikmaschine machen, liegen Computer-Welten. Wohlgemerkt: Was sich heute auf dem Monitor abspielt, wäre technisch auch schon vor einem knappen Jahrzehnt möglich gewesen – nur wußte noch keiner, wie's geht.

Speziell mit Gleichgesinnten oder in User-Gruppen (die Szene) hat man sich zusammengesetzt, um in unzähligen (oft auch vergeblichen) Versuchen die Register des VIC-Chip und der Sprungvektoren ab $0300 (768) zu manipulieren, und so möglichst das Letzte aus diesem 8-Bit-Computer herauszuholen – zumindest in grafischer Hinsicht.

Seit einiger Zeit geistern neue Begriffe durch die Szene, die sich wie Codes aus Geheimakten des Pentagon oder chemischer Verbindungen anhören: FLD, DYSP und DYCP, AGSP usw. Das sind Abkürzungen für trickreiche Spielarten der Grafikprogrammierung, die wir mit Beispielen demonstrieren möchten. Wenn Sie sich selbst an solche Programmobjekte heranwagen, sind Maschinensprache-Kenntnisse (vor allem für Interrupt-Manipulationen) und die nötigen Programmierwerkzeuge (Assembler, Maschinensprache-Monitor) unerläßlich.

Geheimnisse des Rasterzeilen-Interrupts

Alle Effekte beruhen auf dem Prinzip, die Schirmanzeige durch den VIC bei jedem Bildaufbau (25 x pro s) aufs neue zu beeinflußen: Unterschiedliche Farben, Ausschalten des Bildschirmrands und mehr als acht Sprites gleichzeitig sind nur einige programmtechnische Highlights, die man durch Manipulation des Raster-IRQs erreicht.

Um den Bildschirm makellos zu beeinflussen (ohne Flackern und Flimmern), muß man ungeschriebene Gesetze des Timings einhalten, sprich Manipulationen nur in bestimmten, exakt begrenzten Phasen des Bildaufbaus zulassen.

Diese VIC-Speicherstellen liefern Rückmeldungen über die Tätigkeit des Chips:

– Register 18 ($D012) und Bit 7 von Register 17 ($D011): die Nummer der aktuellen Rasterzeile,

– Register 19 ($D013) und 20 ($D014): Positionen in x- und y-Richtung, bei der ein Lightpen-Impuls stattfand,

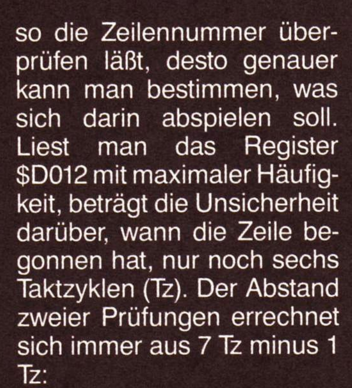
– Register 30 ($D01E) und 31 ($D01F): Sprite-Kollisionsflag.

Bei entsprechender Programmierung kann der VIC-Chip überprüfen:

- ob die gewünschte Rasterzeile erreicht ist,
- ob es einen Lightpen-Impuls gab oder
- eine Kollision zwischen zwei Sprites oder mit dem Hintergrund stattfand.

Diese Ereignisse können eine Interrupt-Anforderung an den Mikroprozessor weiterleiten und damit ein Programm initialisieren, das keine Rechenzeit mit der Prüfung einzelner Register verschwendet. Die Rückmeldung deren Zustände wird in Register 25 ($D019) gespeichert.

Am besten eignet sich die Nummer der augenblicklichen Rasterzeile zur Bestimmung der aktuellen Bildaufbauphase. Wenn ein Programm diese Speicherstelle liest, erfährt es zwar die Zeilennummer, nicht aber, was in der Rasterzeile gezeigt werden soll. Je öfter man also die Zeilennummer überprüfen läßt, desto genauer kann man bestimmen, was sich darin abspielen soll. Liest man das Register $D012 mit maximaler Häufigkeit, beträgt die Unsicherheit darüber, wann die Zeile begonnen hat, nur noch sechs Taktzyklen (Tz). Der Abstand zweier Prüfungen errechnet sich immer aus 7 Tz minus 1 Tz:



```
lda #zeile
LABEL WARTE:
cmp reg18 ;4 Tz
bne label warte ;2/3 Tz
```

Programmiert man den VIC so, daß er einen Raster-IRQ auslösen soll, geschieht das exakt zu Beginn der gewünschten Zeile. Doch die CPU nimmt die Interrupt-Anforderung erst dann an, wenn der momentan gültige Befehl abgearbeitet ist. Da die Zeitdauer von Assemblerbefehlen zwischen zwei und sieben Taktzyklen variiert, kann sich die Akzeptanz des IRQ bis zu 6 Tz verzögern. Diese Verzögerung ist scheinbar zufällig. Doch läßt sich eine während des Programmablaufs gleiche, von Start zu Start jedoch variierende Verzögerung einbauen. Dazu müssen nach einer bestimmten Zeit wieder dieselben Befehle abgearbeitet werden (oder solche mit denselben Zyklen!): periodischer Aufbau, sagt der Fachmann. Man muß den Zeitaufwand der Interrupt-Routine und den des Hauptprogramms so angleichen, daß die Zeit für einen Bildschirmaufbau (19 656 Tz) minus Zeitbedarf der IRQ-Routine ein Vielfaches der Periodendauer des Hauptprogramms beträgt:



$$19656 - I = k * P$$

Während des Programmablaufs wird immer derselbe Befehl im gleichen Stadium unterbrochen. Nach einem Neustart kann das aber ein anderer Befehl bzw. ein anderes Stadium sein. Mit Ausprobieren und Berechnen läßt sich letztendlich eine Zeitbalance schaffen, um z.B. Flimmern bei Farbumschaltungen zu vermeiden. Dieses Gleichgewichtsverhältnis ist äußerst empfindlich: Schon ein Tastendruck oder Bildschirmscrolling erzeugt einen flimmernden Screen.

Wichtig für eine einigermaßen exakte Reaktion auf Interrupts ist, daß ihre Annahme vom Hauptprogramm nicht verhindert wird. Gibt's dennoch zwingende Gründe dafür, muß die Interrupt-Routine eventuelle Verzögerungen abpuffern, die dadurch entstehen. So kann man z.B. auf eine spätere Rasterzeile warten und den Interrupt auslösen, bevor sie erreicht wird. Voraussetzung: Der Abstand sollte natürlich entsprechend der Dauer des Interrupt-Verbots genügend groß gewählt werden.

Bei manchen Effekten kann man auf den IRQ-Puffer verzichten, wenn man weiß, wie die aktuelle Bild-

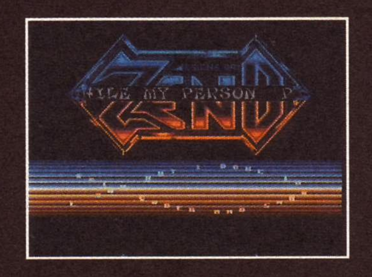
Hexenküche der Intro-Programmierer

EFFEKTHASCHEREI

Ein gutes Computerspiel ohne ausgefuchstes Intro ist für viele wie eine Suppe ohne Salz. Wie kommt es, daß Sprites, Zeichen und bunte Rasterzeilen so munter über den Bildschirm tanzen? Wir haben Profis über die Schulter geguckt.







aufbauphase aussehen soll. Beispiel: das Ausschalten des senkrechten Rands mit drei Rasterzeilen Toleranz. Umschaltungen an gerade nicht gezeigten Farben akzeptieren Ungenauigkeiten, die vom gepufferten IRQ aufgefangen werden. Bei sichtbaren Farben oder nach dem Wegschalten des seitlichen Bildschirmrands braucht man allerdings ein genaues Zeitverhältnis zum Bildaufbau.

Das folgende Assembler-Listing im Hypra-Ass-Format (dieser Assembler ist im 64'er-Sonderheft 71 ausführlich beschrieben) halbiert die Ungenauigkeiten bei jedem Rasterzeilenwechsel. Beachten Sie dazu unser Assemblerlisting auf Seite 39 dieses Heftes.

Zuerst programmiert man den Interrupt so, daß die Ungenauigkeit unter 8 Tz liegt. Dann wartet die Routine so lange, bis bei einem mittleren Verzögerungswert (im Vergleich zum Idealfall) der nächste Zeilenwechsel stattfände. Jetzt wird geprüft, ob die tatsächliche Verzögerung kurz oder lang ist. Bei einer kurzen Unterbrechung

Professionelle Vorspänne

muß man die Dauer der mittleren Verzögerung addieren (4 Tz). Kurze (0 bis 3 Tz) und lange (4 bis 7 Tz) Verzögerungswerte werden überlagert (0+4=4, 1+4=5, 2+4=6, 3+4=7). Mit drei aufeinanderfolgend angepaßten Halbierungen korrigiert man so die maximale Ungenauigkeit von 7 Tz. Jetzt liegt das Zeitverhältnis des Interrupt-Programms zum Bildaufbau eindeutig fest. Wenn das Listing nach dem Assemblieren als Objekt-Code vorliegt, kann man es mit »SYS 36864« initialisieren und »SYS 36867« wieder abschalten.

Auf der Sonderheftdiskette finden Sie vier Intro-Demos, untermalt von fetzigem Sound, die wir näher unter die Lupe nehmen wollen. Diese Programme sind gepackte Dateien – das Entpacken dauert einige Sekunden.

F. L. D (Flexible Line Distance)

Dieser, unter Profis altbewährte Grafiktrick, gibt jeder Bildschirmrasterzeile einen variablen Abstand zur vorhergehenden. Das geht natürlich nur über die Abfrage des Raster-IRQs. Unser Beispiellisting, das Sie mit jedem Assembler eingeben und in ein Objektfile umwandeln können, läuft im Interrupt, d.h., es wird jede 1/60 s aufgerufen. Die Routine muß die Arbeit des VIC zwischen zwei Bildschirmzeilen kurz verzögern, damit diese einen gewissen Abstand bekommen. Das Programm liest aus der Adresse $D012 die aktuelle Rasterzeile, reduziert sie mit AND #$07 auf einen Wert zwischen 0 und 7 und legt das Ergebnis in Speicherstelle $D011 ab. Es ist das VIC-Register, das u.a. fürs bitweise, vertikale Scrolling des C 64 zuständig ist. Die aktuelle Rasterzeile verschiebt sich jetzt um einen Pixelabstand nach unten. Diese Prozedur wiederholt sich nun entsprechend der Anzahl der Rasterzeilen, die laut Tabelle zwischen der letzten und aktuellen Bildschirmzeile freigelassen werden sollen. Anschließend verzögert eine Leerschleife die Routine so lange, bis der Rasterstrahl acht Pixelzeilen auf den Bildschirm gebracht hat. Dann verzweigt sie zur Bearbeitung der nächsten Rasterzeile an den Routinenanfang. Währenddessen bringt der Schirm die aktuelle Zeile. Zum Schluß ruft die Routine den normalen Systeminterrupt auf. Um den rollierenden Effekt zu erreichen, wählt man für jede Zeile variable Abstandswerte, die zyklisch verändert werden.

Sie müssen die Routine in dieser Form ab Adresse $1000 im Speicher plazieren und per »SYS 4096« starten. Sehen Sie selbst, wie munter es jetzt auf dem Bildschirm zugeht! Selbstverständlich lassen sich die Tabellenwerte zum Experimentieren verändern.

Laden Sie unser Programmbeispiel auf der Sonderheft-diskette mit:

```
LOAD "1-PIXEL FLD",8
```

und starten Sie mit RUN.

Das Demo stammt von der Gruppe »Zone 45« und benutzt die Regeln unserer vorgestellten Routine. Man sieht das deutlich am Textscrolling im mittleren Bildschirm (Abb. 1). Nach dem Entpacken liegt der Demotext ab Adresse $3604, vor RUN belegt das Programm den Bereich von $0801 bis $3524. Nach einem Reset ist der Neustart mit »SYS 2051« möglich.

Hier unser Assembler-Listing:

```
sei ;Interrupt sperren
lda #$7f
sta $dc0d
lda #$f1
sta $d01a
lda #$1b
sta $d011
lda #$2e
sta $d012
lda #<main ;IRQ-
Vektoren auf eigene Routine
sta $0314  ;richten
lda #>main
sta $0315
lda #$01 ;Laufvariablen
vorbereiten
sta $03
sta $02
cli ;Interrupt-
Sperre aufheben
rts
MAIN: lda #$01
sta $d019 ;IRQ-
Flag loeschen
ldy $02
LINE: ldx #$00
DIST: lda $d012
LOOP1: cmp $d012 ;noch
gleiche Zeile?
beq loop1 ;wenn nein, dann-
 weiter!
and #$07
adc #$18
sta $d011 ;verknuepfen und-
 in $d011 ablegen
inx
txa
cmp tabelle,y ;Vergleich
mit tabelle
bne dist ;wenn nein, dann
zurueck
tya ; Y retten
pha
bit $ea
nop
lda #$00
LOOP2: nop
ldy #$08
LOOP3: dey
bpl loop3
inx
cpx #$07
bne loop2 ;Verzoegerung um-
 8 Pixelzeilen
pla
tay ; Y zurueckholen
iny
inc $03
lda $03
cmp #$10 ;Anzahl der
Durchlaeufe/Zeilen
bne line ;erreicht?
lda #$00
sta $03
inc $02 ;wenn ja, dann
Tabellenzeiger um 1
lda $02 ;erhöhen
cmp #$20 ;schon alle
Werte?
bne end
lda #$00
sta $02

END: jmp $ea31 ;wenn ja, zum System-Interrupt
TABELLE:  .byte $01,$01,$02,$02,$03,$03,$03,$04
          .byte $04,$04,$05,$05,$05,$05,$06,$06
          .byte $06,$06,$05,$05,$05,$05,$04,$04
          .byte $04,$03,$03,$03,$02,$02,$01,$01
          .byte $01,$01,$02,$02,$03,$03,$03,$04
          .byte $04,$04,$05,$05,$05,$05,$06,$06
          .byte $06,$06,$05,$05,$05,$05,$04,$04
          .byte $04,$03,$03,$03,$02,$02,$01,$01
```

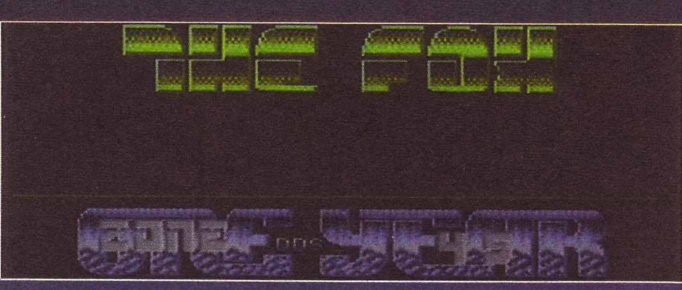


[2] Schachbretter im Rasterzeilen-Interrupt: Flexgrid



[3] Es sieht nur so aus, als wär's ein einziger Buchstabe: die tanzenden Buchstaben des »Plexer DYCP«

Etwas anders, aber dennoch nach demselben Prinzip arbeitet das nächste Beispielprogramm auf Diskette: Flexgrid. Es wurde von der Gruppe »Fairlight« entwickelt und benutzt den Multicolormodus. Im oberen Bildschirmteil erkennt man schachbrettförmige Gitter (Grids), die im wechselnden Rhythmus flexibel ihre Form verändern (Abb. 2). Zwei Rasterbereiche, die ebenfalls rhythmisch die Farbe wechseln, überlagern die Gittermuster horizontal. In der Mitte erscheint Info-Text. Wenn Sie ihn ändern möchten, müssen Sie die Daten ab $9037 mit neuem Text belegen. Das Ergebnis können



[1] Verschiebt die Laufschrift um jeweils ein Pixel: der FLD-Effekt

Sie sich mit »SYS 2101« ansehen.

DYCP (Different Yielding Character Position)

Zu diesem Spezialeffekt gibt's ebenfalls ein Beispiel auf der Sonderheftdiskette:

```
LOAD "PLEXER DYCP",8
```

Nach Laden startet man mit RUN. Wenn die Datei entpackt ist, beginnt die Hauptroutine ab $1000.

Der mathematische Begriff Sinus bringt's auf den Punkt: Es geht in diesem Demo vor allem um die sinusförmige Auf- und Abbewegung von Zeichen (Characters), die über farbige, statische Rasterzeile im unteren Bildschirm huschen (Abb. 3). Demoprogrammierer müssen fleißige Tipper sein: Alle Sinus-Positionen sind in einer Tabelle abgelegt, aus der sich das Programm die Positionsdaten holt. Auffällig ist, daß sich zwei scheinbar unabhängige Laufschrift-Zeilen über den Schirm bewegen (Double-DYCP). Das sieht aber nur so aus: Wenn man sich den Speicher ab $110F betrachtet (dort steht der Text für die untere Laufschrift), erkennt man, daß hier einzelne Zeichen für beide Texte abwechselnd hintereinander stehen: erster Buchstabe der ersten Textzeile, dann erster Buchstabe von Text 2 usw.

Ein weiteres Demo, das diese Grundlagen verwendet (DYSP, Different Yielding Sprite Position) haben wir ausführlich im 64'erMagazin 1/92, Seite 50, vorgestellt. Programmiert wurden beide Intros von 64'er-Redakteur Peter Klein.

Bob Plotting

Auch diese Routine läßt sich wie ein Basic-Programm laden und starten:

```
LOAD "BOB PLOTTING",8
RUN
```

Dieses Programmbeispiel wurde von den »Dark Mights« entwickelt und verzichtet auf Farbspielereien. Wer schon in Basic versucht hat, Sinuswellen im Hires-Bildschirm zu erzeugen, weiß, wie lange das dauert. Daß es mit Maschinensprache viel schneller geht, ist nichts Neues. Interessant ist aber, daß man hier für die

PLOT-Routine (Setzen eines Pixels im Hires-Bildschirm) den Rasterzeilen-Interrupt benutzt. Die Sinus-Werte werden ebenfalls nicht berechnet (das dauert auch in Assembler zu lang), sondern aus einer chronologischen Tabelle gelesen. Damit läßt sich eine Unmenge verschiedener Sinus-Kurven erzeugen (Abb. 4). Bis zur Animation ist's nur noch ein kleiner Schritt: Um das wilde Flackern beim ständigen Bildschirmlöschen zu unterdrücken, wurde das Problem durch »Double-Buffering« gelöst.

Zur Demonstration, daß man per Sinus-Tabelle mehr als nur eine Kurve erscheinen lassen kann, sollten Sie folgende Tasten ausprobieren (sonst bleibt der Bildschirm dunkel):

<A> bis <Z>: ruft 26 verschiedene Bewegungsabläufe auf.

<1> bis <9>: steigende Animationsgeschwindigkeit,

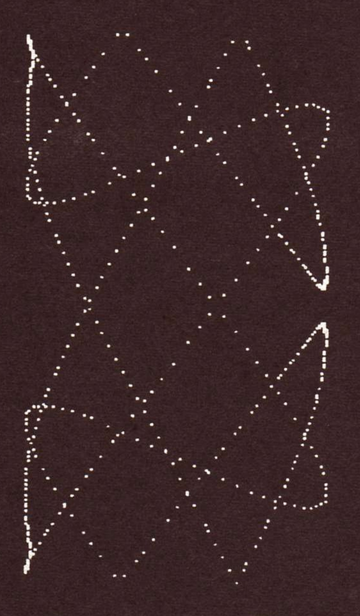
<0>: unterbricht die Bewegung auf dem Bildschirm,

<RESTORE>: Programmende.

Nach einem Reset läßt sich das Programm mit »SYS 4096« erneut starten. Die Speicherbereiche von $2000 bis $3FFF und $4000 bis $5FFF werden als Bitmaps benutzt.

Es gibt noch weitere Tricks, um die Bildschirmausgabe zu manipulieren. **AGSP**, ein Verfahren zum Bildschirm-Scrollen, verwendet jeweils nur **eine** vertikale Pixelreihe. Welche gerade an der Reihe ist, wird durch einen Offset in VIC-Register 17 ($D011) festgelegt. Maximal sechs Bildschirme lassen sich beliebig nach links und rechts scrollen – sonst würde der höchstmögliche Inhalt von Adresse $D011 den Wert »255« übersteigen. Diese Methode ist allemal schneller, als jedesmal 64 000 Pixel (= 1 Bitmap) zu verschieben.

Einfaches Scrollen verschiebt nur einen Bildschirminhalt in bestimmter Richtung und mit festgelegter Geschwindigkeit. Beim **Parallax-Scrolling** (Mehr-Ebenen-Bildschirmrollen) bewegen sich zwei oder mehrere, übereinanderliegende Bildschirme (Playfields) mit unterschiedlichem Speed in verschiedene Richtungen. Dort, wo eine Ebene durchsichtig ist, erkennt man den darunterliegenden Bildschirm (z.B. Berge hinter Bäumen). Dieser Effekt läßt sich mit Glasplatten vergleichen, die mit verschiedenen Motiven bemalt sind und in unterschiedlicher Geschwindigkeit hintereinander verschoben werden (Abb. 5): Die Tiefenwirkung ist verblüffend. Manfred Trenz z.B. (»Katakis«, »Turrican 1 und 2«) ist Experte auf diesem Gebiet. Wenn Sie die Texte der entpackten Intros ändern möchten (man findet die Speicherbereiche nach dem Entpacken mit jedem vernünftigen Maschinensprache-Monitor), sollten Sie die veränderte Gesamtdatei nach dem erneuten Speichern wieder komprimieren: Ungepackt sind Sie oft länger als 100 Blocks.



[4] Fast unbegrenzte Möglichkeiten mit einer einzigen Sinustabelle: Bob Plotting

Wer Assembler beherrscht, sollte sich die Demolistings mit einem geeigneten Maschinensprachemonitor ansehen (erst nach dem Entpacken!). Die verwendeten Routinen lassen sich leicht analysieren. Damit erhält man wertvolle Anregungen, um selbst solche Projekte in Angriff zu nehmen. (bl)

```
0     -; RASTER-MASTER 0.11 31.10.1988
1     -; (C) 1988 BY  T C
2     -;
3     -.GL BILD       = $1B    ;AN
4     -.EQ ZEILE1     = $26
5     -.EQ ZEILE2     = ZEILE1+2
6     -.EQ NORMAL     = 14
7     -;
8     -.GL IRQVECT    = $0314
9     -;
10    -.GL VICCTRL1   = $D011  ;VIC-CONTROL-REG.
11    -.GL RASTER     = $D012
12    -.GL IRQFLAGS   = $D019
13    -.EQ IRQMASKS   = $D01A
14    -.EQ BORDER     = $D020
15    -;
16    -.EQ CIA1ICR    = $DC0D
17    -;
18    -.EQ OLDIRQ     = $EA31
19    -.EQ IRQRET     = $EA7E
20    -;
21    -;***** MAKROS *****
22   .-.MA SETIRQ (ZEILE)
23    -           LDA #BILD!O!((ZEILE/2)!A!$80)
24    -           STA VICCTRL1
25    -           LDA #<(ZEILE)
26    -           STA RASTER
27    -           LDA #$FF        ;LOESCHEN
28    -           STA IRQFLAGS
29    -.RT
30    -;
31    -.MA SETVECT (ROUT)
32    -           LDA #<(ROUT)
33    -           LDY #>(ROUT)
34    -           STA IRQVECT
35    -           STY IRQVECT+1
36    -.RT
37    -;
38    -;+++++ STARTADRESSE +++++
39    -;
40    -           .BA $9000
41    -;
42    -           JMP INIT
43    -           JMP AUS
44    -;
45    -;***** VORBEREITUNG *****
46    -INIT       SEI
47    -; TIMER-IRQ VERHINDERN
48    -           LDA #$7F
49    -           STA CIA1ICR
50    -; RASTER-IRQ ERLAUBEN
51    -           LDA #$01
52    -           STA IRQMASKS
53    -;
54    -           ... SETVECT(ROUT)
55    -           ... SETIRQ(ZEILE1)
56    -           CLI
57    -;
58    -           RTS
59    -;
60    -;***** ENDE *****
61    -AUS        SEI
62    -; RASTER-IRQ VERHINDERN
63    -           LDA #$00
64    -           STA IRQMASKS
65    -; TIMER-IRQ ERLAUBEN
66    -           LDA #$81
67    -           STA CIA1ICR
68    -           ... SETVECT(OLDIRQ)
69    -;
70    -           CLI
71    -           RTS
72    -;
73    -;***** IRQ-_OUTINEN *****
74    -;
75    -TOOLATE    JMP IRQRET
76    -;
77    -;----- GRUENER STRICH = 'RASTER-MASTER'
78    -;
79    -ROUT       LDA #<(ZEILE2)
80    -           CMP RASTER
81    -           BCC TOOLATE
82    -           BEQ TOOLATE
83    -;
84    -WZEILE     CMP RASTER
85    -           BNE WZEILE
86    -;
87    -           LDX #10
88    -WARTE1     DEX
89    -           BNE WARTE1
90    -;
91    -           NOP
92    -           LDA RASTER      ;59-66
93    -           CMP #<(ZEILE2+1)
94    -           BEQ OK1
95    -           BIT $AA
96    -           NOP
97    -; 68-71
98    -OK1        LDX #9
99    -WARTE2     DEX
100   -           BNE WARTE2
101   -;
102   -           NOP
103   -           NOP
104   -           NOP
105   -           LDA RASTER      ;124-127
106   -           CMP #<(ZEILE2+2)
107   -           BEQ OK2
108   -           BIT $AA
109   -; 131-132
110   -OK2        LDX #10
111   -WARTE3     DEX
112   -           BNE WARTE3
113   -;
114   -           NOP
115   -           LDA RASTER      ;188-189
116   -           CMP #<(ZEILE2+3)
117   -           BNE OK3
118   -; 194!
119   -;---------
120   -OK3        LDX #3
121   -WARTE4     DEX
122   -           BNE WARTE4
123   -;
124   -           NOP
125   -           NOP
126   -           NOP
127   -           LDA #0
128   -           STA BORDER
129   -           LDA #NORMAL
130   -           STA BORDER
131   -;
132   -           LDA #$FF
133   -           STA IRQFLAGS
134   -           JMP OLDIRQ
135   -;
136   -;+ + + + +  E N D E  + + + + +
```

Das Source-Listing von »Raster Master« im Hypra-Ass-Format

Hi-Eddi - der Zeichenprogrammklassiker

Zeichen- und Malprogramme für den C64 gibt's inzwischen in Hülle und Fülle. Ein Grafik-Tool setzte Maßstäbe und Standards, die noch heute gelten: Hi-Eddi – zumal wir uns intensiv den großartigen Animationsmöglichkeiten dieses Zeichenprogramms widmen.

Schwarzweiß oder Farbe? Das entscheiden Sie. »Hi-Eddi« bietet den gesamten Bildschirm Ihres Monitors als Zeichenfläche und kann alle Grafiken ausdrucken.

Hier eine Übersicht der wichtigsten Eigenschaften:
- Es läßt sich wahlweise auch als farbiges Malprogramm verwenden, obwohl es den Multicolormodus nicht aktiviert,
- sieben Bereiche für Schwarzweißgrafiken, sechs für Farbbilder stehen zur Verfügung,
- integrierter Sprite-Editor mit komfortablen Funktionen (spiegeln, drehen usw.). Außerdem lassen sich Sprites aus einer Hires-Grafik kopieren oder dort unterbringen.
- Mit einer Zeichentrickfilm-Funktion kann man flimmerfreie Bewegungsabläufe bis zu 24 Bilder/s programmieren.
- Mehrere Grafikbildschirme lassen sich neben- und untereinander ausdrucken. Das ergibt Superbilder mit einer Breite von 640 Punkten und unbegrenzter Länge.

Laden Sie das Zeichenprogramm mit:

```
LOAD "HI-EDDI",8
```

dann starten Sie mit RUN.

Wenn der Maschinensprache-Teil »Hi.Exe« nachgeladen ist, meldet sich das Programm mit der Frage nach der Betriebsart. Falls Ihnen der Schwarzweißmodus genügt, können Sie hier »0« eingeben (oder einfach nur <RETURN> drücken). Möchten Sie farbige Bilder entwerfen, muß die Eingabe »128« lauten. In beiden Fällen meldet sich der Grafikbildschirm 1 (ab $2000) mit dem kreuzförmigen Zeichencursor, der per Joystick in Port 2 gesteuert wird.

Funktionen

Die einzelnen Zeichenmodi aktiviert man per Tastendruck:

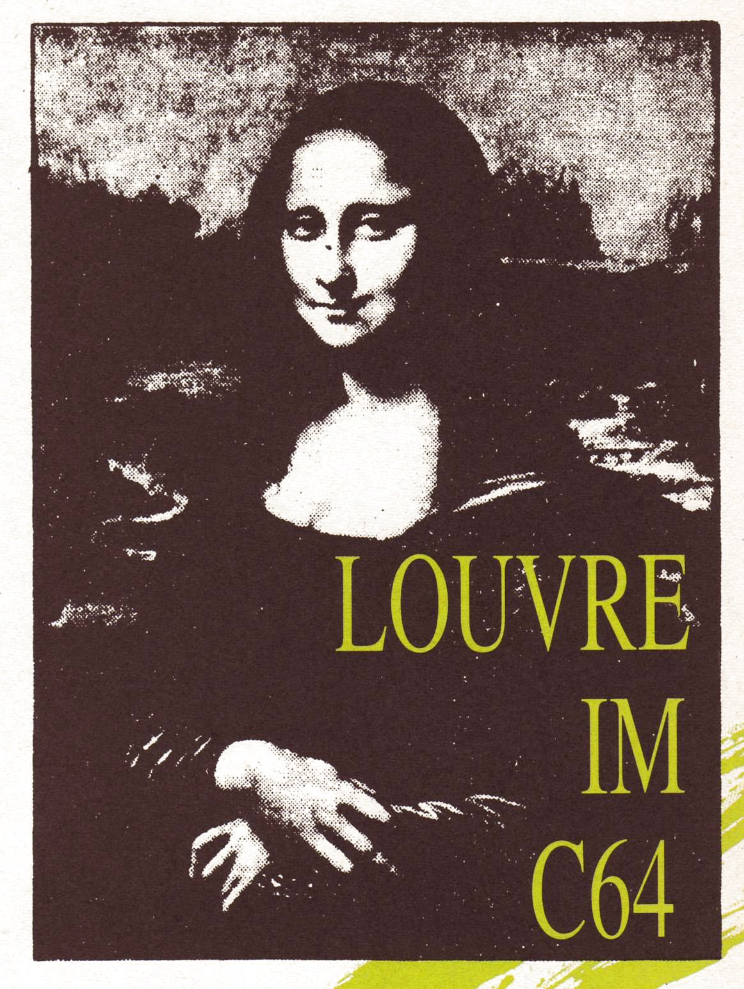
<D> **Draw**: Freihändig zeichnen. Der Grafikcursor läßt sich mit dem Joystick über die Bildfläche bewegen. Wenn Sie den Feuerknopf drücken, wird ein Pixel an der aktuellen Cursorposition gesetzt. Um Punkte zu löschen, muß zusätzlich die SHIFT-Taste gedrückt werden.

<L> **Line**: Linien ziehen. Mit dem ersten Druck auf den Feuerknopf legt man den Anfangspunkt einer durchgehenden Linie fest, mit dem zweiten den Endpunkt, ein dritter Tipp auf »Feuer« fixiert wieder einen Anfangspunkt usw. Den jeweiligen Linienbeginn kann man mit <F7> speichern. So lassen sich z.B. strahlenförmige Gebilde erzeugen (Abb. 6). Gelöscht wird mit zusätzlichem Druck auf <SHIFT>.

<R> **Rectangle**: Rechtecke entwerfen. Der erste Knopfdruck legt die linke obere Ecke, der zweite die rechte untere des Rechtecks fest, das dann auf dem Bildschirm erscheint. Die Tasten <F7> und <SHIFT> wirken wie bei <D> und <L>.

<C> **Circle**: Kreise zeichnen. Der erste Druck auf den Feuerknopf legt den Kreismittelpunkt fest, der zweite einen beliebigen Randpunkt. Das Programm zieht die Kreise im Uhrzeigersinn.

Reicht der Radius über die Bildschirmränder hinaus, bricht die Funktion ab. Möchte man Kreissegmente zeichnen, muß der zweite Knopfdruck einen Punkt am Bildschirmrand markieren. Der sichtbare Ausschnitt erscheint jetzt auf dem Bildschirm. Achtung: Der Radius darf 256 Pixel nicht überschreiten!



<P> **Paint**: Ausmalen begrenzter Flächen. Dazu setzt man den Grafikcursor in den auszufüllenden Bereich und drückt den Feuerknopf. Wenn in der Umrandung ein Loch ist, färbt sich auch der übrige Bildschirm. Dann muß man die Aktion mit erneutem Knopfdruck abbrechen. Wie man Füllflächen wieder löscht, steht in der Beschreibung zu <I> (Invertieren).

<M> **Move**: Bildschirmbereiche verschieben. Wie bei <R> sucht man per Feuerknopf zwei diagonale Ecken des zu transponierenden Bereichs. Beim zweiten Knopfdruck wird er farblich markiert. Der dritte Knopfdruck teilt »Hi-Eddi« die linke, obere Ecke des Zielbereichs mit. Achten Sie darauf, daß dieser Grafikausschnitt vollständig in den gesamten Bildschirm paßt. Er darf aber den Quellbereich überlappen oder in einem anderen Bildschirm liegen. Die Auflösung des Move-Befehls entspricht dem normalen Textbildschirm: Es lassen sich nur Grafikausschnitte verschieben, die dem 40 x 25-Byteraster entsprechen.

Wollen Sie denselben Grafikbereich noch an andere Stellen verschieben, genügt ein Druck auf <Pfeil links>. Damit wird die letzte Markierung des Quellbereichs wieder aktiviert. Lediglich die neuen Zielkoordinaten muß man bestimmen. Achtung: Das gilt nur, wenn man zwischenzeitlich nicht den Zeichenmodus wechselt!

<T> **Text**: Buchstaben und Grafiksymbole einfügen. Auf dem Zeichenbildschirm erscheint ein 8 x 8-Pixel großer Cursorrahmen an der mit dem Joystick gewählten Position. Jetzt können Sie alle Tastaturzeichen eintippen: Der Text wird in die Grafik integriert (Abb. 2). Der Cursorrahmen reagiert fast wie im Textmodus: Steuerung per Cursortasten, Löschen mit <DEL> und Einfügen (<SHIFT DEL>. Nur <RETURN> zeigt keine Wirkung. Wenn Sie den Rahmen an anderer Stelle plazieren möchten, stehen lediglich die Cursortasten oder der Joystick zur Verfügung.

Wenn Sie im »T«-Modus zu den übrigen Zeichenbefehlen umschalten möchten, müssen Sie zusätzlich die CTRL-Taste drücken (z.B. <CTRL D>. Um den Textmodus anschließend wieder aufzurufen, ist erneut <T> zu drücken.

<G> **Get Sprite**: Sprite aus Grafik entnehmen. Der Rahmencursor nimmt jetzt die Größe eines Sprites an. Bei Knopfdruck übernimmt man diesen Bereich als Spritemuster. Jetzt geht »Hi-Eddi« automatisch in den Append-Modus (s. Beschreibung). Sie können nun das Sprite an beliebiger Stelle auf dem Grafikschirm ablegen oder mit dem integrierten Sprite-Editor bearbeiten.

<A> **Append**: Sprite in die Grafik kopieren. Per Feuerknopf fügt man den Spriteinhalt in die Grafik ein. Der darunterliegende Ausschnitt wird nicht gelöscht. Wenn man bei gedrücktem Knopf den Joystick bewegt, entpuppt sich das Sprite als Malerpinsel.

<S> **Stamp**: Sprite auf Bildschirm kleben. Diese Funktion arbeitet wie <A>, allerdings wird der darunterliegende Bereich vorher gelöscht.

<E> **Erase**: Teile des Grafikbildschirms löschen. Der Cursorrahmen wird zum Radiergummi: Er löscht bei gedrücktem Feuerknopf jeden Bereich, den er überfährt.

<F> **Foreground**: Vordergrundfarbe

<B> **Background**: Hintergrund

Diese beiden Anweisungen sind nur wirksam, wenn »Hi-Eddi« im Farbmodus verwendet wird (Betriebsart 128 oder 192). Im Gegensatz zu Multicolor-Malprogrammen (z.B. Amica Paint, Paint Magic usw.) kann »Hi-Eddi« nur Vorder- und Hintergrundfarbe anbieten. Dafür steht aber nach wie vor die volle Auflösung von 320 x 200 Pixeln zur Verfügung (bei Multicolor reduziert sie sich auf 160 x 200!). Bei der Vordergrundfarbe gilt der eingestellte Farbwert (Code 0 bis 15) jeweils für eine 8 x 8-Pixel-Matrix. Ab dem neunten Grafikpunkt läßt sich aber schon wieder eine andere Farbe verwenden. Voraussetzung: Man sollte Koordinaten wählen, die sich ohne Rest durch »8« teilen lassen. Sonst überlappt bei ungeraden Werten die vorher eingestellte Farbe den nächsten 8 x 8-Pixel-Bereich. Grund: Das Farb-RAM zur Hires-Grafik (ab $0400) bietet wie im Textmodus ebenfalls nur 1000 Byte, um die gewünschte Farbe abzulegen. Wenn Sie aber zwischen jedem 8 x 8-Feld genügend Platz frei lassen, können Sie ohne weiteres farbenprächtige Bilder entwickeln - ohne Einschalten des Multicolormodus (Abb. 3).

<F> und <B> erhöhen einerseits die Rahmenfarbe, anderseits wählt man damit den Fore- bzw. Background-Modus. Per Knopfdruck nehmen jetzt die gesetzten Pixel (=Vordergrund) oder gelöschte Bildpunkte (=Hintergrund) die aktuelle Rahmenfarbe an. Dies geschieht aber nur immer im 8 x 8-Pixelbereich, auf dem sich der Grafikcursor gerade befindet.

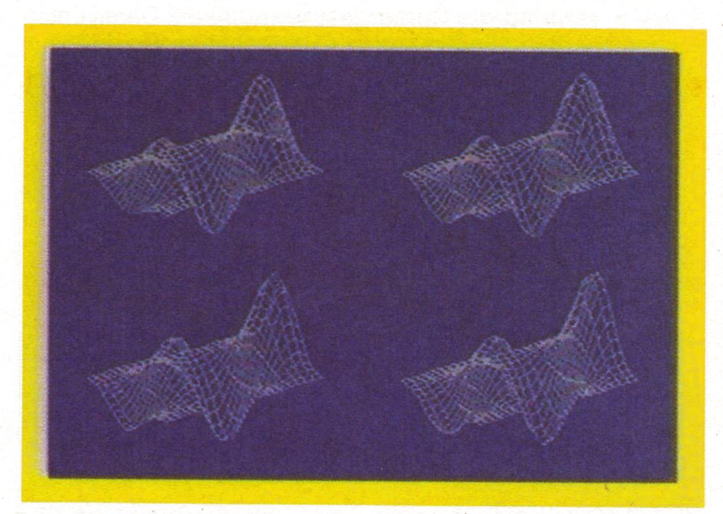
Wurde ein Feld zuviel angepinselt, muß man die SHIFT-Taste gleichzeitig mit dem Feuerknopf drücken: Die vorherige Farbe erscheint wieder an dieser Stelle. Die Taste <Pfeil links> aktiviert die alte Farbe ebenfalls, aber für den gesamten Zeichenbildschirm.

Direktbefehle

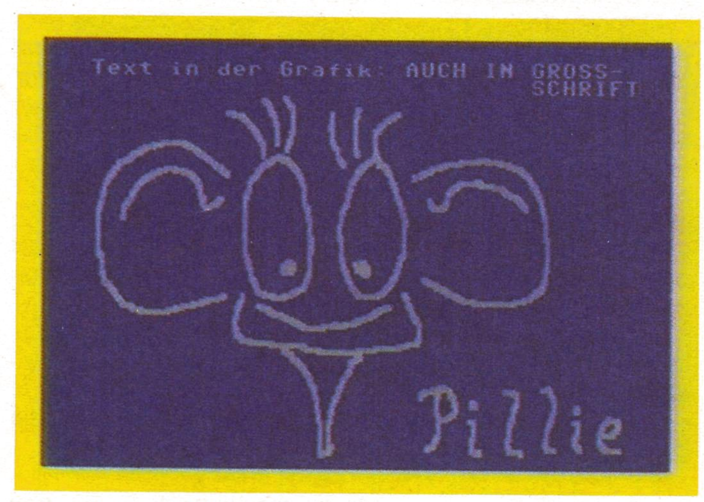
Die bisher genannten Tastenfunktionen veränderten die einzelnen Modi des Zeichenprogramms. Die folgenden wirken innerhalb der gewählten Funktionsarten:

<1> **bis** <7> **Hires-Bildschirm wählen**: In der Schwarzweißbetriebsart (0) gibt's sieben, beim Farbmodus sechs Hires-Bereiche, die auf den Monitor geholt werden. Ihre Verwendung ist vielseitig: Man kann beispielsweise Zwischenstadien einer Grafik speichern und wieder bei einer älteren Version beginnen, wenn man bei der neueren etwas vermurkst hat. Ebenso lassen sich Konstruktionssets entwickeln, z.B. eine Anzahl von Symbolen für Schaltpläne. Per Sprite-Befehl baut man die Schaltungssymbole dann im Nu in den Hauptbildschirm ein.

<I> **Invertieren**: zeigt den gesamten Hires-Schirm revers: Vorder- und Hintergrundfarben werden vertauscht. Diese Anweisung ist äußerst nützlich, wenn bei <P> (Paint) Farbe durch ein Loch in der Umrandung geschlüpft ist und sich auf dem übrigen Bildschirm ausbreitet. Per Feuerknopf verhindern Sie größeren Schaden und stoppen die virenartige Verbreitung der Farbpixel. Jetzt wird der gesamte Bildschirm mit <I> invertiert. Lassen Sie jetzt das entstandene Loch vollaufen und drücken Sie erneut die Taste <I>: Alles ist wie vor dem Farbunfall!



[1] Im »Moviecreator« kann man 15 verschiedene Bewegungabläufe auswählen



[2] In Wort und Bild: erläuternder Text in der Grafik

<**U Schirmnummer**> **AND-Verknüpfung**: Eine weitere sinnvolle Anwendung der sieben Hires-Bereiche: Der sichtbare Schirm wird mit der (unsichtbaren) Hires-Grafik eines der übrigen Screens verbunden. Das Ergebnis erscheint im sichtbaren Hires-Bereich.

<**O Schirmnummer**> **OR-Verknüpfung**: Diese Funktion eignet sich z.B. zum Duplizieren von Bildern.

<**X Schirmnummer**> **Exor-Verbindung**: Wenn Sie vorher zwei Grafiken durch die Befehle U oder O gemischt haben, können Sie das Original mit zweimaliger Eingabe von »X nummer« wieder herstellen.

<**SHIFT B**> **Total Background**: Wie im Farbbetrieb von »Hi-Eddi« muß zuerst die gewünschte Farbe mit der Taste <B> eingestellt sein (im Bildschirmrahmen sichtbar). Sie

wird dann der Hintergrundfarbe angepaßt, wenn man gleichzeitig die SHIFT-Taste drückt.

<**SHIFT F**> **Total Foreground**: stellt die Pixelfarbe der gesamten Grafik auf den vorher mit <F> gewählten Wert ein.

Die beiden letztgenannten Funktionen agieren auch bei Schwarzweiß ebenso wie im Farbmodus! Mit <Pfeil links> lassen sich auch hier Irrtümer korrigieren.

<**SHIFT CLR**> **Hires-Schirm löschen**: Alle aktuellen Grafikpixel werden vom Bildschirm verbannt, die Hintergrundfarbe bleibt aber erhalten.

< **+** > **schneller Grafikcursor**: Das Cursorkreuz bewegt sich per Joystick zunächst langsam und steigert seine Geschwindigkeit proportional zur zurückgelegten Strecke.

<**–**> **verminderte Cursorgeschwindigkeit**: Damit unterbinden Sie den zunehmenden Cursor-Speed. Pixelgenaue Arbeit am Grafikbild wird damit viel leichter.

Die Funktionstasten haben eine doppelte Aufgabe zu bewältigen:

<**SHIFT und F1, F3, F5, F7**> **Koordinatentabulator**: speichern die jeweiligen Cursorpositionen. Dazu muß gleichzeitig die SHIFT-Taste gedrückt werden. Das Programm merkt sich jetzt eine der vier möglichen Standorte des Grafikcursors. Wenn Sie nun die entsprechend belegte F-Taste ohne SHIFT drücken, springt das Cursorkreuz exakt an die vorbestimmte Stelle. <F7> wird von den Befehlen <L> (Line), <R> (Rectangle) und <C> (Circle) automatisch beansprucht.

Die andere Funktion der F-Tasten:

<**CBM + F1, F3, F5, F7**> **Cursorsprung einstellen**

<**H**> **horizontale Schrittweite**, <**V**> **vertikaler Abstand**: In der Normaleinstellung bewegt sich der Grafikcursor in 1-Pixel-Schritten, bzw. um acht Bildpunkte per Cursortasten. Diese Abstände lassen sich aber frei programmieren. Vier Schrittweitenpaare in x- und y-Richtung können mit den Funktionstasten <F1> bis <F7> gespeichert werden. Drücken Sie die gewünschte F-Taste gleichzeitig mit <CBM> und wählen Sie anschließend den Buchstaben für die Richtung (<H> oder <V>). Die neuen Werte lassen sich jetzt eingeben. Achtung: Diese Funktion gilt nur für die Steuerung per Cursortasten; der Joystick holt seine Schrittweite immer aus <F1>.

Nach dem Einschalten sind die F-Tasten bereits vorbelegt:

- <F1>: H = 1, V = 1. Durch Erhöhung der Werte auf »2« oder »3« lassen sich im DRAW-Modus punktierte Linien zeichnen.
- <F3>: H = 8, V = 8. Speziell im TEXT-Modus <T> lassen sich so größere Buchstaben- oder Zeilenabstände einstellen.
- <F5>: H = 24, V = 21. Stellt die Sprite-Maße ein, die dann im Sprite-Editor gelten.
- <F7>: H = 160, V = 96. Damit unterteilt man den Bildschirm in gleichgroße Viertelbildabschnitte (z.B. für den WALK-Befehl).

Maßstabsgetreue und symmetrische Zeichnungen, Gitterraster usw. können mit Hilfe der F-Tasten im Handumdrehen realisiert werden.

Sprites und Animation

»Hi-Eddi« besitzt einen Sprite-Editor.

<**SPACE**> **Sprite-Entwurfsblatt aktivieren**: Mit der Leertaste ruft man den Editormodus auf. Links oben erscheint ein blinkender Cursor, den man per Joystick bewegt. Per Feuerknopf wird ein cursorgroßer Punkt im Spritemuster gesetzt; mit gleichzeitigem Druck auf SHIFT wieder gelöscht. Wie das entstehende Sprite in Originalgröße aussieht, zeigt die rechte Bildschirmseite (Abb. 4).

Die Befehlstasten des Grafik-Editors stehen im Sprite-Modus nicht zur Verfügung – dafür gibt's andere:

<**M**> **Mirror**: Sprite senkrecht spiegeln.

<**T**> **Turn**: Drehung um 180 Grad. <M>, anschließend <T> ergibt eine waagrechte Spiegelung.

<**R**> **Rotate**: Das Sprite wird um 90 Grad im Uhrzeigersinn gedreht. Da so ein Kobold aber 24 Punkte breit und nur 21 Pixel hoch ist, gehen dabei die drei rechten Sprite-Spalten verloren. Übrigens: 2 x <R> ist nicht dasselbe wie <T> (probieren Sie's aus!)

<**G**> **Grid**: Auf Wunsch läßt sich zum besseren Abzählen von Sprite-Punkten auf der Editorfläche ein Raster einblenden. Ein weiterer Tipp auf <G> löscht das Gittermuster.

Kurzinfo: Hi-Eddi

Programmart: Zeichenprogramm
Laden: LOAD "HI-EDDI",8
Starten: nach dem Laden RUN eingeben
Steuerung: Tastatur/Joystick Port 2
Besonderheiten: Animationssequenzen, Erzeugen von verschiedenfarbiger Grafik, integrierter Sprite-Editor
Benötigte Blocks: 63
Programmautor: Hans Haberl

<**SHIFT CLR**>: Editorfläche löschen.

Durch erneuten Druck auf <SPACE> können Sie den Sprite-Editor wieder verlassen.

Mit »Hi-Eddi« kommt Bewegung in die Grafik! Zwei Befehle stehen zur Verfügung, um schnelle Grafik-Animationssequenzen (keine Sprites!) ablaufen zu lassen:

Neue Zeichensätze entwerfen

<**W**> **Walk: Ablauf einer Bildfolge**: Damit werden die Bildschirmspeicher in schneller, programmierbarer Reihenfolge zyklisch aufgerufen. Allerdings sind sechs Hires-Schirme mit jeweils 8000 Byte ziemlich langsam, um eine ansprechende Animation auf den Screen zu bringen. Speicher 7 dient als Kinoleinwand und wird ständig überschrieben. Das wären lediglich sieben Bilder pro Sekunde. Daher muß man die sechs übrigen Bildschirme in 24 Viertelbilder (jeweils 160 x 96 Pixel) zerlegen (s. Taste <F7> bei der Einstellung der Grafikcursor-Schrittweite und MOVE-Befehl). Diese werden dann bei Maximalgeschwindigkeit in 1 s durchgerasselt. Das entspricht der Ablaufgeschwindigkeit eines Filmprojektors.

<**SHIFT W**> **Bildfolge programmieren**: Dazu muß man eine Sequenz-Zeichenkette vorgeben. Die Zahlen »1« bis »6« bedeuten die Nummern der Hires-Schirme (Voreinstellung); Buchstaben von A bis X rufen die Viertelbilder auf. Wie's genau geht, zeigt unsere Beschreibung zum Hi-Eddi-Demoprogramm »Moviemaker«. Der Filmablauf wird anschließend mit <W> gestartet.

Achtung: Wenn Sie »Hi-Eddi« im Farbmodus betreiben, stehen nur fünf Hires-Bildschirme zum Animationsentwurf zur Verfügung, Nr. 6 dient als Leinwand!

Die Methode von »Hi-Eddi«, den Commodore-eigenen Zeichensatz zu verändern, weicht von den üblichen Zeichensatz-Editoren erheblich ab, ist aber ebenso effektiv: Man muß dazu die Funktionen des Sprite-Editors verwenden.

<**Z**> **Zeichensatz einblenden**: Alle 256 Zeichen des Character-ROMs erscheinen oben auf dem Grafikbildschirm. So läßt er sich ändern und mit beliebigen, auch fremdartigen Zeichenmustern versehen:

1. Wählen Sie <G> – der Grafikcursor in Größe eines Sprites taucht auf.

2. Lenken Sie ihn mit dem Joystick an die Stelle, an der die zu verändernden Zeichen stehen.

3. Per Knopfdruck wird dieser Bereich als Sprite übernommen.

4. Mit <SPACE> ruft man jetzt den Sprite-Editor auf und gibt den gewünschten Zeichen neues Aussehen (z.B. Umlaute oder Sonderzeichen).

5. Speichern Sie den geänderten Zeichensatz z.B. mit der Endung ».ZS« auf Diskette.

Achtung: Die Änderung wirkt sich nicht unmittelbar aus. Die anderen Zeichen gelten erst dann im aktuellen Grafikmodus (z.B. bei <T>), wenn der veränderte Zeichensatz erneut geladen wurde.

Diskettenkommandos

Mit Hilfe der CBM-Taste lassen sich die wichtigsten Diskettenoperationen einleiten:

<CBM L> Load: lädt gewünschte Dateien in »Hi-Eddi«. Per Tastendruck bestimmen Sie, um welche Art es sich handelt:
- <G> = Schwarzweißgrafik,
- <F> = Farbbild,
- <S> = Sprite-Muster,
- <Z> = Zeichensatz.

Anschließend muß man den vollständigen Dateinamen eintragen, den man beim Speichern gewählt hat. Es lassen sich auch Bilder fremder Zeichenprogramme laden. Beachten Sie, daß »Hi-Eddi« keine automatischen Endungen vergibt - weder beim Laden noch beim Speichern. Zur besseren Unterscheidung der Dateien auf Diskette schlagen wir folgende Ergänzungen vor:
- .PIC: Grafik,
- .ZS: Zeichensatz,
- .SPR: Sprites.

Selbstverständlich ist jedes andere Suffix (Endung) möglich.

»Hi-Eddi«-Betriebsarten

Bit	Eingabe	Funktion
7	0	Schwarzweißbetrieb
	128	Farbe
6	0	Eingabe über Tastatur
	64	per Joystick über Menü
5	bis 1	ohne Funktion
0	0	löscht beim Start alle Bildschirmspeicher
	1	ohne löschen

<CBM S> Save: speichert eine Datei auf Diskette. Bei der Angabe von Typ und Namen gelten dieselben Regeln wie bei <CBM L>.

<CBM D> Directory: bringt das aktuelle Disketten-Inhaltsverzeichnis auf den Bildschirm.

<CBM C> DOS-Kommando: Ohne OPEN und CLOSE können Sie der Diskettenstation alle gewünschten Disk-Befehle mitteilen (z.B. S, N, C, V usw.). Es gelten dieselben Syntax-Vorschriften wie beim normalen Floppybetrieb (s. Handbuch).

»Hi-Eddi« besitzt einen separaten Druckertreiber: Hi-Print.

<CBM P>: aktiviert das Nachladen von »Hi-Print« im Overlay-Verfahren. Achten Sie darauf, daß Ihr Drucker eingeschaltet ist - der Ausdruck beginnt unmittelbar danach.

Sie haben mehrere Druckmöglichkeiten:

1. Grafikdruck groß oder klein,

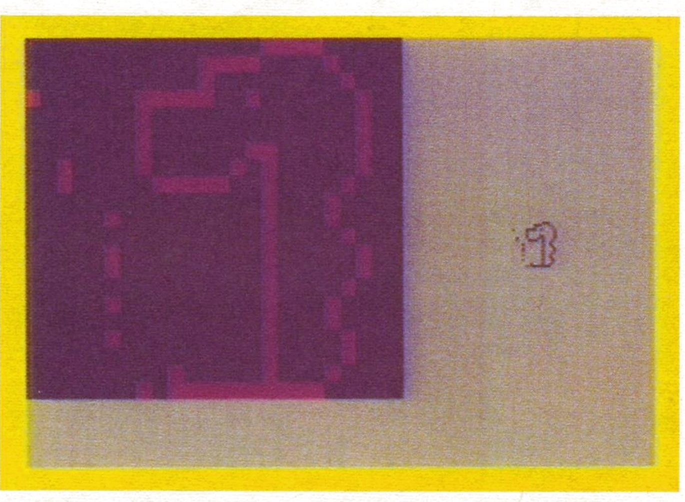
2. Zwei Bilder nahtlos nebeneinander (dann müssen Sie aber vorher die Einstellung »klein« wählen).

Vor und nach dem Ausdruck werden keine zusätzlichen Zeilenvorschübe (Line-Feeds) ausgegeben; man sollte also vorher das Papier entsprechend einrichten. Andererseits lassen sich damit aber alle Bildschirme übergangslos untereinander ausgeben (Abb. 5) - das entspricht einer Super-Hardcopy von 640 x 600 Bildpunkten! Lädt man zwischendurch weitere Bilder von Diskette nach, steht meterlangen Hardcopies nichts mehr im Weg.

Das alte Lied: Drucker ist nicht gleich Drucker. Diese Routine eignet sich nur für Epson-Geräte (die 640 Pixel horizontal



[3] Wenn sie geschickt verteilt werden, sind 16 verschiedene Farben in einem Hires-Bild möglich



[4] Kein separates Programm: Der Sprite-Editor ist integriert.

drucken können) und kompatible mit entsprechendem seriellem Hardware-Interface (egal, ob intern oder extern am Drucker angebracht). So eines ist allerdings in der Druckerroutine auch softwaremäßig vorhanden: Wenn Sie ein Parallelkabel besitzen, so bringen Sie es am Userport des C64 und am Centronics-Anschluß des Druckers an. »Hi-Eddi« erkennt diese Konfiguration automatisch und druckt die Grafik ebenso tadellos wie auf seriellem Datenweg.

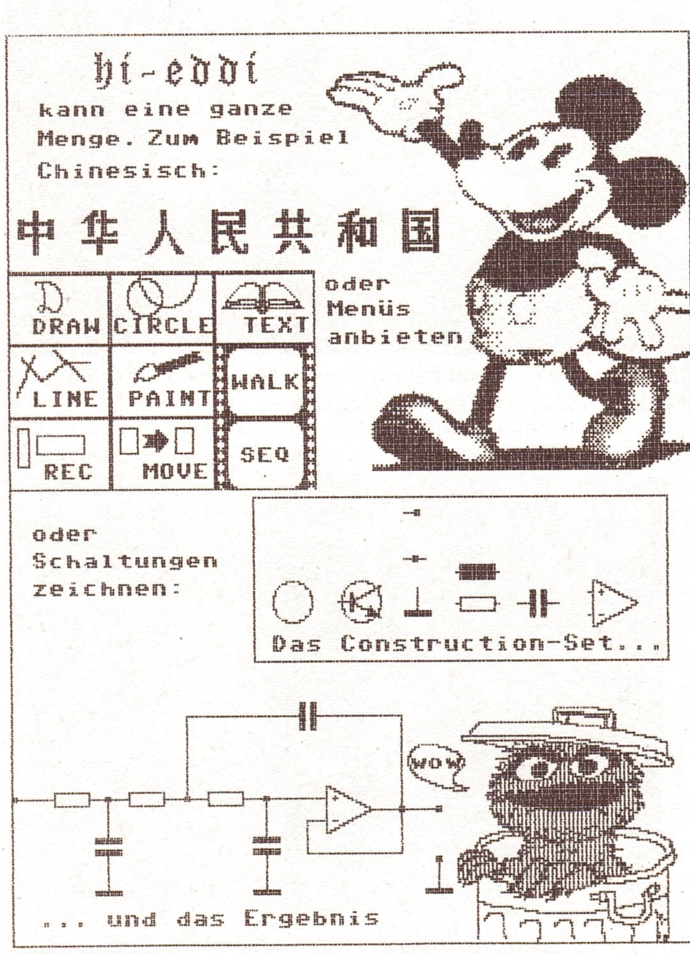
Wenn Sie eine andere Druckerroutine verwenden oder selbst eine programmieren möchten: Sie muß exakt innerhalb des Speicherbereichs von $0D00 bis $1F7F liegen, um keine Daten von »Hi-Eddi« zu überschreiben!

»Hi-Print« belegt den Speicher der Adressen $0D00 (3328) bis $0E7C (3708). Hier die markanten Speicherzellen für

Druckerwerte und Steuerbytes, die sich mit jedem Maschinensprache-Monitor ändern lassen:

- $OD40 (3392) und $0E6A (3690): Geräteadresse 4 (Drucker),
- $0E6B (3691): Sekundäradresse (Voreinstellung: 1). Damit ist der Linearkanal des Druckers gemeint. Falls Ihr Gerät oder das Hardware-Interface einen anderen Wert brauchen, müssen Sie den in dieser Speicherstelle eintragen.
- $0E6C (3692) und $0E6D (3693): Wagenrücklauf (CR, $0D) und Zeilenvorschub (LF, $0A)
- $0E6F (3695) bis $0E73 (3699): ESC 3, setzt den 216tel-Inch-Abstand für die Pixelzeilen. Voreingestellt ist $17 (23/216 Inch).
- $0E74 (3700) bis $0E78 (3704): ESC *, aktiviert den 8-Nadel-Einzelbitmodus. Die Voreinstellung berücksichtigt den Wert 4 (80 Dots/Inch, CRT-Grafik).
- $0E79 (3705) bis $0E7C (3708): ESC 2, schaltet den Default-Wert des Zeilenabstands wieder ein (1/216 Inch)

»Hi-Eddi« ist ein zweckorientiertes Anwendungsprogramm



[5] »Hi-Print« fügt zwei Grafiken nahtlos neben- und untereinander. Werden zusätzliche Bilder nachgeladen, sind im Prinzip endlose Druckausgaben möglich.

und sein Autor haßt offensichtlich zeitraubende Eingaben über Windows, Icons und Gadgets: Eine oder maximal zwei Tasten zu drücken geht beim C64 schneller, als per Joystick oder Maus einen Zeiger über den Bildschirm zu bewegen und die gewünschte Funktion exakt an der richtigen Stelle anzuklicken. Beim AT, Amiga oder MacIntosh ist es natürlich was anderes. Wer auch beim C64 nicht darauf verzichten will (weil er's z.B. von Geos so gewohnt ist), kann »Hi-Eddi« auch als Menüprogramm starten (Abb. 8) und die wichtigsten Funktionen auf dem Bildschirm wählen (DRAW, CIRCLE, TEXT usw.). Diskettenoperationen (DIR, LOAD und SAVE) lassen sich in der untersten Menüleiste aktivieren.

Verschiedene Betriebsarten von Hi-Eddi

Zu Programmbeginn ist bei der Frage nach der Betriebsart »192« einzugeben. Unsere Tabelle gibt Auskunft über die Bitbelegung der Speicherstelle, die Ihre Eingabe speichert, und die entsprechenden Werte, die Sie eintragen müssen. Es sind auch Bitkombinationen möglich (z.B. Menü mit Joystick, Farbe, ohne Löschen der alten Bitmap: 128 + 64 +1 = 193).

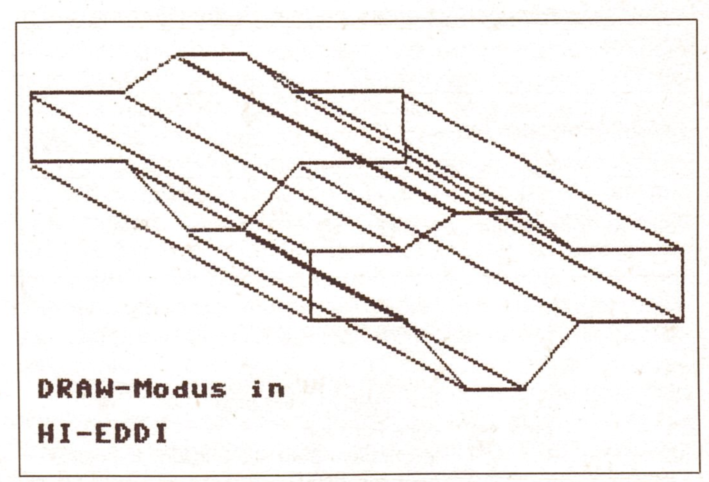
Selbstverständlich bietet die Menütafel auch Vorteile: Die Wahl der Farben ist nicht mehr so umständlich (Erhöhung der Farbwerte durch Weiterschalten per Tastendruck). Sie lassen sich nun mit dem Feuerknopf sofort bestimmen (Brush Foreground/Background, Total Foreground/Background). Auch die Verknüpfungsanweisungen U, O, und X kann man jetzt direkt aufrufen (AND, OR und EXOR). In den Sprite-Editor kommt man z.B. sofort nach Anklicken des entsprechenden Fensters.

Hi-Eddi als Grafikdieb

Haben Sie schon mal eine Grafik aus einem anderen Programm (z.B. ein Spiel) ausgefiltert? Dazu brauchen Sie kein separates Utility. Da »Hi-Eddi« sieben Hires-Bereiche verwalten kann, gibt es keine noch so versteckte Grafik, die das Grafikprogramm nicht aufspüren und übernehmen kann.

Laden Sie ein beliebiges Programm, das die Hires-Grafik enthält, an der Ihnen so viel liegt. Falls sich die fremde Software mit <RUN/STOP RESTORE> unterbrechen läßt - umso besser, sonst müssen Sie den Resetknopf Ihres C64 drücken.

Jetzt sind Sie wieder im Direktmodus und können »Hi-Eddi« laden. Geben Sie nun bei »Betriebsart« die Zahl 1 ein (Grafik nicht löschen). Mit den Zahlentasten 1 bis 7 lassen



[6] Mit der LINE-Funktion lassen sich beliebige Koordinaten verbinden. Unerwünschte Striche kann man im ERASE-Modus wieder von der Bildfläche verschwinden lassen.

sich alle Hires-Speicher aufrufen - die gesuchte Grafik ist garantiert in einem der Bereiche zu finden. Mit <CBM S> kann man das Bild jetzt im Hi-Eddi-Format speichern. Allerdings werden eventuelle Farben nicht berücksichtigt - es funktioniert nur als Schwarzweißgrafik.

Stichwort Farben: Um colorierte Bilder zu speichern, müßte man zwei Dateien auf Diskette anlegen: die Hires-Grafik von $2000 bis $03F3F und das Farb-RAM von $0400 bis $07E7. »Hi-Eddi« transportiert die Farbspeicher-Daten intern ans Ende einer Farbgrafik (ab $4000) und macht eine Gesamtdatei daraus. Deshalb benötigen Farbbilder 37 Blöcke auf einer Diskette. Nach dem Laden wird der Bereich von $4000 bis

$47E7 wieder ins Farb-RAM nach $0400 geschoben.

Außer den Programmdateien HI-EDDI, HI-EXE und HI-PRINT finden Sie eine Menge Grafikbeispiele sowie Hilfs- und Demoprogramme auf der Diskette zu diesem Sonderheft.

»Grafiklader« lädt Schwarzweiß- und Farbbilder ohne »Hi-Eddi«. Allerdings kann man die Bilder nur betrachten, aber nicht bearbeiten. Nach dem Laden des Utilities mit:

```
LOAD "GRAFIKLADER",8
```

und dem Start mit RUN fragt das Programm nach dem Dateinamen des Bildes und lädt es. Auf dem Bildschirm kann man den Ladevorgang verfolgen. Wem die Vordergrund- (0 = schwarz) und Hintergrundfarben (1 = weiß) nicht gefallen, kann das in Zeile 550 des Basic-Programms jederzeit ändern.

»Menue« ist die Hires-Grafik zur bereits beschrieben Farbmenütafel.

»Demo1.Pic« und »Demo2.Pic« sind zwei Schwarzweißgrafiken, die sich nahtlos untereinander drucken lassen.

»Mot1.Pic« bis »Mot6.Pic« sind Einzelgrafiken, die mit Hilfe des WALK-Befehls den Bewegungsablauf eines Viertakt-Ottomotors simulieren. Um diese Animation auszuprobieren, müssen Sie »Hi-Eddi« im Schwarzweißbetrieb aktivieren (Betriebsart 0). Jetzt schalten Sie per Zahlentaste <1> den ersten Hires-Schirm ein und laden mit der Funktion <CBM L> die erste Grafik: Mot1.Pic (vorher <G> für »Grafikbild« eingeben). Drücken Sie anschließend die nächste Zahl (<2>). In diesen Bereich wird jetzt »Mot2.Pic« geladen. Das Spiel geht weiter bis zum Hires-Bereich <6>, der mit »Mot6.Pic« gefüllt wird (Abb. 7). Rufen Sie nun die Funktion <SHIFT W> auf. Den Vorschlag für den Sequenzstring»123456« (= sechs Bildschirme) können Sie unverändert übernehmen (RETURN-Taste drücken!). Mit <W> startet man jetzt die Animation. Auf dem Bildschirm sieht man, wie ein Ottomotor arbeitet: Ansaugen, Verdichten, Zünden und Ausstoßen. Wenn Sie meinen, daß es genug sei, drücken Sie den Feuerknopf - Sie sind dann wieder im Zeichenmodus.

Last not least bietet unsere Sonderheft-Diskette noch ein Programmpaket, das animierte 3 D-Grafik auf den Bildschirm bringt: Moviecreator.

Demoprogramme und -bilder

Formatieren Sie zunächst eine Datendiskette, um die einzelnen Animationsgrafiken zu speichern. Laden Sie nun das Programm mit:

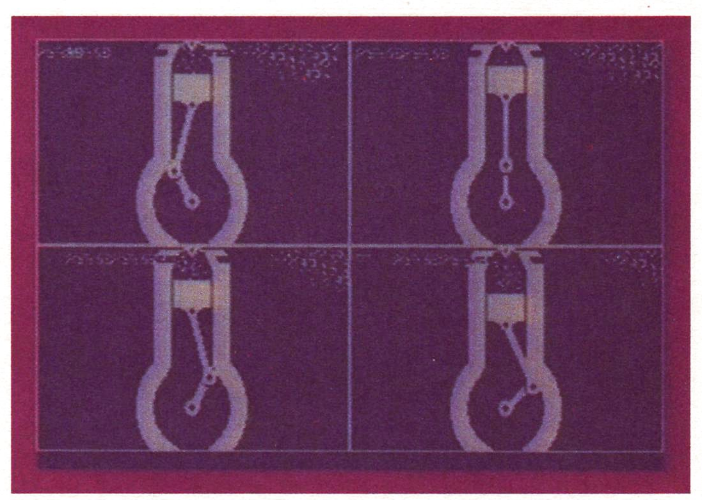
```
LOAD "MOVIECREATOR",8
```

Nach dem Start mit RUN lädt das Basic-Programm die Maschinensprache-Teile »Grapher« (erzeugt die erforderlichen Grafiknetzgitter) und »BMC« (zum Speichern komprimierter Grafiken). Anschließend bietet der Bildschirm 15 Möglichkeiten für Bewegungsabläufe (Abb. 1), dahinter steht in Klammern die Anzahl der benötigten Bitmaps.

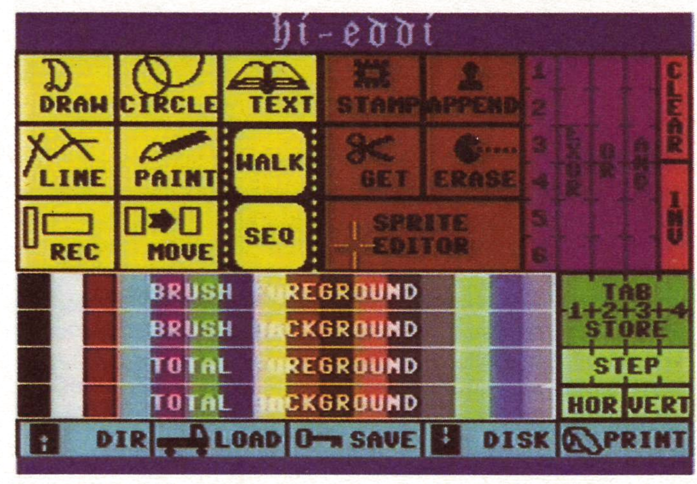
Bevor Sie die gewünschte Zahl eingeben, sollten Sie die formatierte Datendiskette ins Laufwerk legen. Bei der Frage »BMC (j/n)« können Sie entscheiden, wie die einzelnen Bitmaps auf Diskette gespeichert werden sollen: normale Größe (32 Blöcke) oder verdichtet (unter Umständen spart man mehr als 15 Blöcke). Beachten Sie aber, daß komprimierte Grafiken erst wieder entpackt werden müssen, bevor man sie in »Hi-Eddi« laden und weiterbearbeiten kann. Beim Programm »Moviecreator« geschieht dies automatisch, da »BMC« aktiv ist.

Der Hires-Bildschirm wird in vier Viertelbereiche geteilt, in denen jeweils ein Grafikbild erscheint (erzeugt durch »Grapher«). Anschließend wird der gesamte Hires-Bildschirm auf Diskette gespeichert. Pro benötigter Bitmap fügt das Programm die entsprechenden Zahlen an. Wurde der letzte Grafikbildschirm generiert und gespeichert, gibt »Moviecreator« den benötigten Sequenzstring aus, den Sie sich notieren sollten. Statt aus Zahlen besteht er jetzt aus den Buchstaben A bis X. Damit weiß »Hi-Eddi«, daß beim WALK-Befehl keine Gesamtbildschirme, sondern Viertelgrafiken gezeigt werden sollen. Diese erscheinen dann in der Bildschirmmitte. Sie können jetzt eine weitere Animation zusammenstellen (J) oder mit <N> das Programm beenden.

Um sich die bewegte 3D-Grafik anzusehen, müssen Sie erneut »Hi-Eddi« laden und die Grafikbilder, wie beim Viertaktmotor beschrieben, in die diversen Speicher holen und mit <SHIFT W> den vorher notierten Sequenzstring eingeben (überschreiben Sie getrost die Zahlen 1 bis 6). <W> startet den Bewegungsablauf. Wenn Sie dabei die SHIFT-Taste drücken, läuft das Ganze in entgegengesetzter Richtung ab. Mit dem Feuerknopf verläßt man die Animation.



[7] Teilgrafik einer Animationssequenz: Mot6.Pic. Mit den anderen Bildern simulieren Sie einen Viertaktmotor.



[8] Alle Zeichenfunktionen aktiviert man im Menü mit dem Feuerknopf. Beachten Sie, daß nur bei Betriebsart 192 das Menü geladen und aktiviert wird.

»Hi-Eddi« benützt keinen Resetschutz und läßt sich jederzeit mit <RUN/STOP RESTORE> abbrechen.

Achtung: Drücken Sie diese Tastenkombination nicht, wenn gerade ein Befehl ausgeführt wird (Zeichnen, WALK, PAINT usw.), sonst stürzt das Programm ab! Dann hilft nur ein Reset oder Abschalten des Computers.

Im Direktmodus können Sie jederzeit mit »RUN 100« oder »GOTO 150« erneut starten.

Nach kurzer Gewöhnungszeit werden Ihnen die Befehle von »Hi-Eddi« in Fleisch und Blut übergehen. Dann zaubern Sie jede gewünschte Grafik Ruckzuck auf Bildschirm und Drucker! (Hans Haberl/bl)

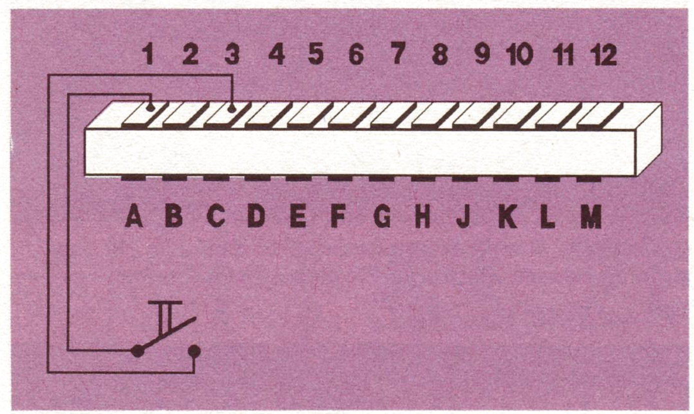
Sir-Freeze - Grafik auf Tastendruck

PRAKTISCHE TIEFKÜHLKOST

Gute Spiele überzeugen auf den ersten Blick durch gelungene Grafiken. Leider muß immer langwierig geladen und gestartet werden. Abhilfe schafft »Sir-Freeze«.

Dieses Tool zaubert Grafik- und Textbildschirme inkl. Sprites auf Diskette. Sie benötigen zur Funktion lediglich einen RESET-Taster (s. u.) und einen Joystick (in Port 2). Die Standbilder lassen sich später von Disk laden und werden einfach mit SYS 4096 auf dem Bildschirm gezeigt. Bevor Sie den Freezer einsetzen muß er unmittelbar geladen werden:

```
LOAD"SIR-FREEZE",8,1
```



RESET am USERPORT (von hinten gesehen)

Danach müssen Sie NEW eingeben und schon wartet Sir-Freeze im Hintergrund auf seine Arbeit.

Sie können jetzt das Programm laden, von dem Sie ein Bild benötigen. Beachten Sie aber, daß der Speicherbereich $8000 bis $8237 nicht überschrieben werden darf, da hier das eigentliche Programm liegt. Wenn Sie dennoch Bilder aus so einem Werk besitzen wollen, gibt es eine Alternative - Sir-Freeze auf EPROM.

Dazu benötigen Sie lediglich einen EPROM-Brenner, ein entsprechendes EPROM (z.B. vom Typ 2732 oder 2764) und ein Modul, mit dem Sie das EPROM für den Speicherbereich $8000 adressieren können. Das File »SIR-FREEZE« ist dafür ohne Änderung geeignet.

Um ein Bild einzufrieren, drücken Sie den Feuerknopf (Joystick in Port 2) und lösen einen RESET aus. Danach sehen Sie auf dem Bildschirm schon Grafikmuster, jedoch meistens nicht das richtige. Die einzelnen Videobänke (es gibt vier) lassen sich mit Joystick links/rechts auswählen. Ist das Bild wieder vollständig sichtbar, sollten Sie Ihre Arbeitsdiskette in die Floppy legen. Es genügt ein erneuter Druck des Feuerknopfs, und das Bild wird unter »PIC (TS)« abgespeichert. Danach führt das Programm einen RESET aus.

Kurzinfo: Sir-Freeze

Programmart: Grafik- und Text-Freezer
Laden: LOAD"SIR-FREEZE",8,1
Vorbereiten: nach dem Laden NEW eingeben und das entsprechende Programm laden
Starten: RESET und Feuer mit Joystick in Port 2
Besonderheiten: läßt sich auch auf EPROM brennen
Benötigte Blocks: 3
Programmautor: Christian Dombacher

Sonderfälle: hier streikt Sir-Freeze

1. Bildschirme mit eingeschaltetem ROM-Zeichensatz

Dieser Fall tritt recht selten auf und kann umgangen werden, indem man den ROM-Zeichensatz mit Hilfe eines Monitors an die Stelle $2000 kopiert.

2. Bildschirme im Raster-IRQ

Wenn der Zeichensatz oder Grafikmodus mit Hilfe eines Raster-IRQ's umgeschaltet wird, wird natürlich nur einer der beiden Bildschirme eingefroren.

Auf Ihrer Arbeitsdiskette läßt sich nicht mehr als ein Bild unter dem gleichen Namen speichern. Sie müssen daher das File umbenennen. Beispielsweise mit:

```
OPEN1,8,15.PRINT#1,"R:PIC 1=PIC (TS)":CLOSE1
```

»PIC 1« ist der neue Name. Ihn können Sie beliebig wählen, dürfen aber die max. Länge von 16 Zeichen nicht überschreiten. Achten Sie nach dieser Aktion auf die Floppylampe. Falls sie blinkt ist etwas schief gegangen.

Aktivieren der gefrorenen Bildschirme

Sie laden zum Betrachten mit

```
LOAD"PIC 1",8,1
```

wobei der Zusatz »,1« wichtig ist. Das so geladene File läßt sich mit SYS4096 starten und macht, wenn alles korrekt verlief, das eingefrorene Bild wieder »mobil«. Es hat folgenden Aufbau:

```
$0E00-$0FFF: Sprites
$1000-$10FF: Aktivierungsroutine
$1100-$16FF: Frei
$1700-$172E: Daten der Videoregister
$172F-$17FF: Frei
$1800-$1BFF: Farbram
$1C00-$1FFF: Bildschirmspeicher
$2000-     : Zeichensatz oder Grafikvorspann
```

Der RESET-Taster

Um einen RESET ausführen zu können, benötigen Sie einen RESET-Taster. Die einfachste Methode ist natürlich einen fertigen zu kaufen. Aber er läßt sich auch leicht selbst basteln:

Auf der Rückseite Ihres C64 befinden sich einige Anschlüsse. Darunter auch der sog. USERPORT. Er stellt die häufigst genutzte Verbindung zu eigenen Basteleien dar. Mit zwei Drahtstücken, und einem Taster (billig in jedem Bastlerladen zu haben), kann man in ca. zehn Minuten einen RESET-Schalter installieren. Das Gehäuse wird vorsichtig geöffnet und auf die obere Seite der Platine wird am hinteren Teil des Pin 1 der eine Draht, an den hinteren Teil des Pin 3 der zweite Draht gelötet. »Hinten« ist wichtig, da Sie ansonsten nie mehr einen USERPORT-Stecker anschließen können. Beachten Sie dabei bitte dreierlei:

1. Beim Öffnen verlieren Sie jeglichen Garantieanspruch.
2. Der C64 muß vom Netzteil getrennt und angeschlossene Geräte müssen ausgesteckt werden.
3. Verwenden Sie einen »Feinlötkolben« von max. 15 Watt.

Wen diese Risiken abschrecken, der hat die Möglichkeit, bei seinem Händler einen USERPORT-Stecker zu erstehen. An diesen werden dann entsprechend der obigen Anleitung die Drähte mit dem Taster gelötet. Die Abb. zeigt die Belegung des Userports mit den gekennzeichneten Anschlüssen. (gr)

Tips & Tools

TIEFSTAPLER MIT PFIFF

Tips & TRICKS

Sieben kleine, aber interessante Routinen haben wir für Sie komponiert. Sie machen Grafik-Handling zum Genuß.

Sie haben es in diesem Heft schon kennengelernt – »Hi Eddi«. Für dieses funktionelle Zeichenprogramm stellen wir die Druckertreiber für den MPS 801/803 und für den MPS 802 vor. Aber auch diejenigen, die Sprites oder sogar ganze Grafikbilder in DATA-Zeilen umsetzen wollen, kommen mit »V.D.E V1« nicht zu kurz. Zur Sprite-Programmierung bieten wir einen Sprite-Sucher und falls Sie eine Bildschirmmaske gestalten möchten, ist »Mask« der ideale Generator dafür. Wem 40 Zeichen am Bildschirm nicht ausreichen, der kann es ja mal mit dem softwaremäßigen 80-Zeichen-Bildschirm versuchen und schließlich finden alle, die keine Lust haben erst lange Programme zu laden, um eine Grafik zu betrachten, »Hilo V3.0«.

Hi-Eddi und MPS 801/803

Also angenommen, neben dem C 64 steht noch der gute alte Drucker MPS 801. Er ist zwar nicht der Schnellste und auch die Beschaffung von Farbbändern ist etwas schwierig geworden (Fa. Boeder stellt sie noch her). Aber er versieht nach wie vor seinen Dienst. Wenn Sie ihm unter Hi-Eddi Grafikdrucke entlocken wollen, sollten Sie die Routine »HI-PRINT /801« zuerst auf Ihre Hi-Eddi-Diskette kopieren. Danach sind nur noch ein paar Handgriffe nötig.

1. Benennen Sie zuerst den originalen Treiber »HI-PRINT« um:

```
OPEN1,8,15.PRINT#1,"R:HI-PRINT /EPS=HI-
PRINT":CLOSE1
```

2. Wenn nach <RETURN> in dieser Zeile der Cursor wieder erscheint, müssen Sie natürlich dem neuen MPS-Treiber den richtigen Namen geben, sonst findet ihn Hi-Eddi nicht auf der Diskette. Vorher sollten Sie aber sicherheitshalber einen Blick auf die Lampe der Diskettenstation werfen. Wenn Sie blinkt, hat irgend etwas nicht geklappt (Schreibschutz auf der Diskette?).

```
OPEN1,8,15:PRINT#1,"R:HI-PRINT=HI-
PRINT /801":CLOSE1
```

Benennt den MPS-Treiber um. Ab jetzt arbeitet Ihr MPS 801 mit Hi-Eddi zusammen. Allerdings mit einer Einschränkung: Ein Nebeneinanderdrucken zweier Bilder oder Groß-Klein-Druck ist nicht möglich.

Trotzdem liefert diese Routine brauchbare Ergebnisse mit dem MPS 801/803 und allen anderen kompatiblen Druckern (Seikosha VC-Serie).

Kurzinfo: Hi-Print 801

Programmart: Druckertreiber für MPS 801/803 und Hi-Eddi
Laden: innerhalb von Hi-Eddi
Besonderheiten: Druckt keine Bilder nebeneinander
Benötigte Blocks: 2
Programmautor: Florian Kulzer

Hi-Eddi und MPS 802

Jeder MPS 802-User kann ein Lied davon singen – ist sein Schmuckstück doch zu keinem Grafikmodus kompatibel. Aber auch für die Besitzer dieser Geräte haben wir einen Treiber. Dazu kopieren Sie von der beiliegenden Diskette die Routine »HI-PRINT /802« auf Ihre Hi-Eddi-Diskette und geben folgende Befehlsfolge ein (fortlaufend in eine Zeile, der Umbruch ergibt sich an einer anderen Stelle):

```
OPEN1,8,15.PRINT#1,"R:HI-PRINT /EPS=HI-
PRINT":CLOSE1
```

Diese Folge benennt den originalen Treiber auf der Arbeitsdiskette um. Wenn der Cursor wieder erscheint, muß der MPS-Treiber den richtigen Namen erhalten, sonst hat Hi-Eddi keine Chance, ihn auf der Diskette zu finden. Werfen Sie vor dem nächsten Arbeitsschritt sicherheitshalber einen Blick auf die Floppylampe.

```
OPEN1,8,15:PRINT#1,"R:HI-PRINT=HI-
PRINT /802":CLOSE1
```

Jetzt stehen Ihnen alle Druckfunktionen von Hi-Eddi zur Verfügung.

Kurzinfo: Hi-Print 802

Programmart: Druckertreiber für MPS 802 und Hi-Eddi
Laden: innerhalb von Hi-Eddi
Benötigte Blocks: 3
Programmautor: Franz Illetschko

Data-Zeilen-Generator

Dieses Tool ist so flexibel, daß Sie es für fast alle Zwecke, in denen DATA-Zeilen erzeugt werden sollen, einsetzen können. »V.D.E. V1« ist die Abkürzung für variabler DATA-Zeilen-Erzeuger Version 1. Die von ihm erzeugten DATA-Zeilen lassen sich in jedem BASIC-Programm übernehmen. Für die Sprite- oder Grafikdaten von Hi-Eddi, sollten Sie in Hi-Eddi mit <RUN/STOP RESTORE> unterbrechen.

```
POKE55,0:POKE56,160:CLR
```

den BASIC-Speicher wieder auf seine Originallänge zurücksetzen. Hi-Eddi verringert in seinem Programm diesen Speicherbereich, um alle Bildschirme und Maschinenprogramme unterzubringen. Wenn Sie den Speicher nicht zurücksetzen, kommt es zu »OUT OF MEMORY ERROR«. Anschließend laden Sie den DATA-Generator mit:

```
LOAD"V.D.E. V1",8
```

und starten mit RUN. Spätestens jetzt sollten Sie wissen welcher Speicherbereich gewandelt werden soll. So steht der Sprite-Inhalt in den Speicherstellen von 704 bis 766 ($02C0 bis $02FE) und die Grafikspeicher ab 8192 ($2000), 16384 ($4000), 24576 ($6000) usw. Sprites haben eine Länge von je 64 Bytes und die Grafikspeicher 8000 Bytes. Es ist allerdings nicht empfehlenswert einen kompletten Grafikspeicher in Datazeilen zu wandeln, da in den meisten Fällen durch die Länge der Datazeilen der Grafikspeicher überschrieben ist (in den DATA-Zeilen ist ein Byte des Grafikspeichers vier Bytes lang). Sie würden also die Datenlänge künstlich auf mehr als das Vierfache vergößern (und ca. fünf Minuten warten). Nehmen Sie daher lieber den Sprite-Inhalt. Achtung! Bei allen Eingaben sind nur ganzzahlige, positive Werte erlaubt und bei Start-Zeilennummer und Schrittweite darf die erlaubte größte Zeilennummer nicht überschritten werden (63999).

Nach dem Programmstart erscheint die erste Abfrage nach der Formel. »V.D.E. V1« gibt hier »PEEK (LN)« vor. »PEEK()« ist dabei die Formel und »LN« wird als Zähl-Variable behandelt. Anfangswert, Schrittweite und Anzahl der Schritte werden später festgelegt. Natürlich läßt sich diese Formel än-

dern, aber gerade hier müssen Sie darauf achten, daß beim Ergebnis nur positive, ganzzahlige Werte erscheinen (z.B. über die INT-Funktion). Wenn Sie die Formel so wie vorgegeben übernehmen, liest PEEK die Werte aus dem Speicher.

Als nächstes geben Sie die Nummer für die erste DATA-Programm-Zeile an (Vorgabe 1000), danach die Schrittweite (Vorgabe 10). Auch sind bei Änderung der Vorgabewerte nur ganzzahlige, positive Werte erlaubt.

Die nächste Eingabe bezieht sich auf den Startwert der Zählvariablen »LN«. Die Vorgabe »0« müssen Sie mit »704« überschreiben. Sonst wäre die erste ausgelesene Speicherstelle »0«. Da wir aber den Sprite ab Speicherposition 704 auslesen wollen, muß hier diese Zahl erscheinen.

»LN« erhöht sich für jedes DATA-Element um die Schrittweite. Sie wird bei der nächsten Abfrage eingetippt. Als Vorgabe erscheint »1«. Bestätigen Sie hier mit <RETURN>. Wenn Sie z.B. mit »2« überschreiben wird nur jede zweite Speicherstelle ausgelesen.

Im letzten Schritt wird die Anzahl der DATA-Elemente verlangt. Überschreiben Sie hier die Vorgabe »100« mit »64«, da das Sprite aus 64 Byte besteht.

Kurzinfo: V.D.E. V1

Programmart: DATA-Generator
Laden: LOAD"V.D.E. V1",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 5
Programmautor: Matthias Strecker

Sprite-Sucher

Damit Sie die mit dem Spritemon (S. 24) konstruierten Animationen auch speichern können, und um nur mal schnell nebenbei ein paar Sprites zu suchen, finden Sie im Sprite-Sucher genau das Richtige (Abb. 1). Durch seine Länge von nur drei Blocks ($0801 bis $0A88) belegt er kaum Speicherplatz. Dadurch ist fast der gesamte Speicher für die Suchfunktionen frei. Geladen wird dieses nützliche Tool mit

```
LOAD"S-SUCHER V2.1",8
```

und gestartet mit RUN. Danach sehen Sie am Bildschirm vier Sprites in zwei Gruppen. Das linke davon steht jeweils für den gerade angezeigten Spriteblock in Multicolor- das rechte in Normaldarstellung. Die Farben lassen sich über die Funktionstasten ändern:

<**F1**> - Multicolorfarbe 1
<**F3**> - Multicolorfarbe 2
<**F5**> - Spritegrundfarbe
<**F7**> - Hintergrundfarbe

In der dritten Zeile steht die Blocknummer. Die aktuelle Speicherposition läßt sich durch die Formel »Blocknummer x 64« berechnen. Die Blocks 32 bis 42 (32 x 64 = 2048 bis 42 x 64 = 2688) sind vom Programm belegt. Per Tastendruck wechseln Sie die Blocks:

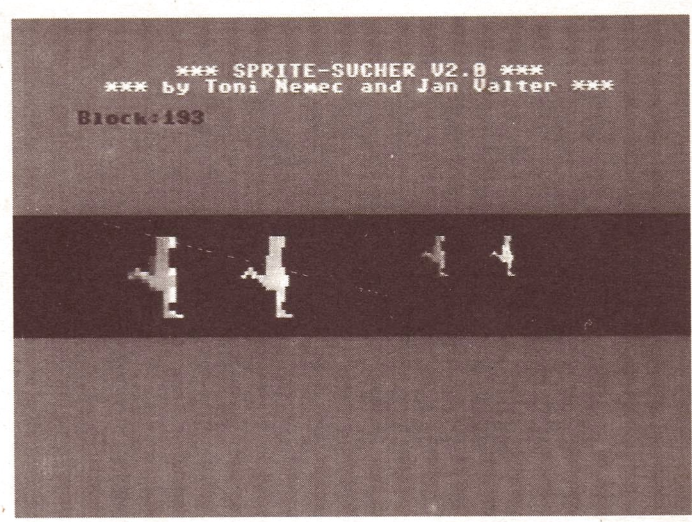
<**CRSR rechts**> - einen Block vor
<**CRSR runter**> - einen Block zurück

Speichern lassen sich Sprites von einem Block Länge bis zu kompletten Sequenzen. Dazu markieren Sie zuerst den ersten Block mit <**RETURN**> und den letzten mit <**S**>. Danach wird immer mit dem Namen »S« gespeichert.

Zusätzlich ist im Sprite-Sucher ein kurzes Programm integriert. Es ermöglicht ein Nachladen von gespeicherten Spritedaten in eigenen Programmen. Dieser LOADER wird, wenn Sie <**L**> drücken, unter dem Namen »L« auf Diskette abgelegt.

Um ihn später zu aktivieren, muß er zuerst geladen werden:

```
LOAD"L",8,1
```



[1] Beim Sprite-Sucher läßt sich fast der gesamte Speicher durchsuchen

Danach sollten Sie NEW eingeben, um die Basic-Pointer zurückzusetzen. Zum Nachladen, müssen die gewünschten Sprite-Daten unter dem Namen »S« auf Diskette gespeichert sein. Sie lassen sich an die Blockpositionen 0 bis 255 laden. Doch achten Sie darauf, daß die Betriebssystem-Pointer nicht überschrieben werden, sonst stürzt der Computer ab. Für die Blockposition ist Speicherstelle 2 im C64 reserviert. Wenn Sie z.B. ab Block 128 laden wollen, geben Sie ein:

```
POKE 2,128: SYS 50944
```

Achtung: Das Betriebssystem der Floppy läßt es nicht zu, mehrere Files gleichen Namens auf Diskette zu speichern. Bevor Sie eine zweite Sprite-Sequenz speichern, müssen Sie daher das Programm mit <**RUN/STOP RESTORE**> unterbrechen und danach das erste File umbenennen. Beispielsweise mit:

```
OPEN1,8,15:PRINT#1,"R:S1=S":CLOSE1
```

Danach läßt sich der Sprite-Sucher mit RUN wieder starten.

Kurzinfo: Sprite-Sucher

Programmart: lädt und speichert Sprites
Laden: LOAD"S-SUCHER V2.1",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: erzeugt externes Ladeprogramm
Benötigte Blocks: 3
Programmautor: Karl Lindfeld

Maskengenerator

Wer programmiert, braucht auf jeden Fall eine Bildschirm-Maske, die seinem Kunstwerk den professionellen Rahmen gibt. Nun ist es aber umständlich, lästig und langwierig, so lange an PRINT-Anweisungen zu basteln, bis endlich ein einigermaßen akzeptables Ergebnis zustande kommt. Mancher Programmierer hat dafür schon mehr Zeit verschwendet als fürs eigentliche Programm. In Zukunft entfällt dieser Ärger. Sie zeichnen einfach Ihre Bildschirmmaske mit allen Texten, Strichen und was es sonst noch gibt, sogar inkl. Farbe. Dann genügt ein Tastendruck, und der komplette Screen wird in PRINT-Anweisungen eingebaut, die sich auf's einfachste in BASIC-Programme einbauen lassen.

Als ersten Schritt laden Sie den Maskengenerator:

```
LOAD"MASK.BSCLDR",8
```

und starten mit RUN. Momente später steht ein Maschinenprogramm im Speicher von $C000 bis $C177. Mit SYS49152 aktivieren Sie es. Danch können Sie beliebig Cur-

sorgrafiken auf dem Bildschirm editieren, allerdings: auf Anführungsstriche, egal ob revers oder normal, muß verzichtet werden. Es kommt sonst später zu Komplikationen in BASIC.

Wenn Sie die Grafik vollendet haben, plazieren Sie den Cursor in die rechte untere Ecke und drücken <F1>. Daraufhin wird ein BASIC-Programm erzeugt, das den Bildschirminhalt als PRINT-Zeilen enthält. Das so generierte Programm beginnt mit der Zeilennummer 0 und jede folgende Zeilennummer ist je um 1 höher. Diese Werte lassen sich vor dem Initialisieren ändern. Nehmen wir an, Sie benötigen Zeilennummer 100 am Anfang:

```
ZN=100:POKE49186,ZN/256:POKE49181,ZN-
PEEK(49186)/256
```

Auch die Schrittweite läßt sich ändern. Allerdings nur bis 255. Wir verwenden als Beispiel »10«:

```
SW=10:POKE49191,SW
```

Ändern Sie einfach die Werte hinter »ZN« bzw. »SW« auf Ihre eigenen Bedürfnisse.

Kurzinfo: Maskengenerator

Programmart: PRINT-Generator für Bildschirmmasken
Laden: LOAD"MASK.BSCLDR",8
Starten: nach dem Laden RUN eingeben und mit SYS49152 initialisieren
Benötigte Blocks: 6
Programmautor: Thomas Förster

80-Zeichen-Bildschirm

Ihr C 64 zeigt leider nur 40 Zeichen in einer Zeile. Für viele Anwendungen ist dies nicht ausreichend. Die Alternative dazu ist »80 Characters« - eine 80-Zeichen-Karte, für die Sie keinerlei Hardware-Erweiterung benötigen. Allerdings sollten Sie einen Monitor angeschlossen haben, da ein normaler Fernseher die hohe Auflösung nicht darstellen kann. Um die Funktionen zu zeigen, befindet sich auf der beiliegenden Diskette ein Demoprogramm:

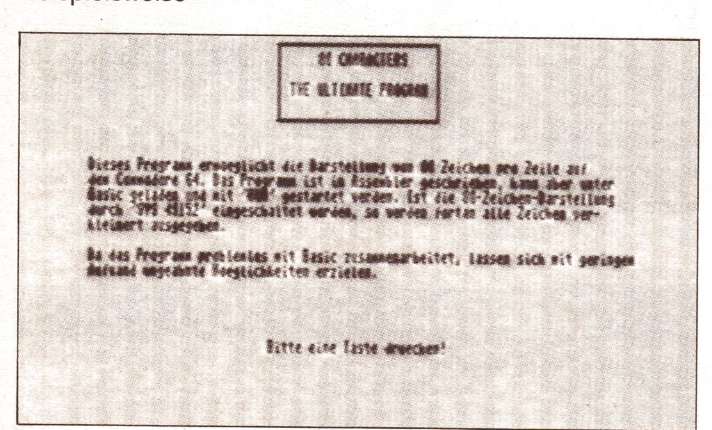
```
LOAD"80-ZEICHEN-DEMO",8
```

und starten Sie es mit RUN (Abb.2). Wenn Sie dieses Programm unterbrechen (<RUN/STOP>) steht in den Zeilen 1 bis 20 der Maschinenspracheanteil, die Zeilen 100 bis 1000 stellen den BASIC-Anteil dar. Für eigene Programmiervorhaben laden Sie:

```
LOAD"80 CHARACTERS",8
```

und schreiben Ihr Programm mit höheren Zeilennummern als 20 dazu.

Der 80-Zeichen-Bildschirm läßt sich nur im Programm-Modus mit SYS49152 einschalten. Um auf die 80-Zeichen-Darstellung umzuschalten, ist daher ein Programm nötig. Beispielsweise



[2] »80 Characters«, ein superkurzes Tool, macht Ihren C-64-Bildschirm (fast) zum PC-Monitor

```
100 SYS49152
110 FORX=0 TO 200:PRINT"80-ZEICHEN-MODUS";
120 NEXT
```

wenn Sie dieses kleine Programm starten, wird 200mal der Text »80-Zeichen-Modus« ausgegeben. »READY« erscheint wieder im 40-Zeichen-Modus. Außerdem läßt sich mit SYS 49339 auf 40 Zeichen zurückschalten und sogar die Farbsteuerzeichen funktionieren in der PRINT-Anweisung. Lediglich »INSERT« und »DELETE« sind gesperrt. Mit PRINT CHR$(14) läßt sich auf Groß/Kleinschrift und mit PRINT CHR$(142) auf Groß/Grafik umschalten. Nach Ausführung eines Programms oder »STOP« wird immer in den 40-Zeichen-Modus gewechselt.

Kurzinfo: 80 Zeichen

Laden: LOAD"80 CHARACTERS",8
Starten: innerhalb eines Programms mit SYS49152
Benötigte Blocks: 7
Programmautor: Marcel Sommerik

Grafik und Text

Dieses sieben Blocks kurze Programm ist ein perfektes Tool zum Laden, Betrachten, Beschriften und Speichern von HIRES-Grafiken (Abb. 3). Geladen wird es mit

```
LOAD"HILO V3.0",8
```

und gestartet mit RUN. Wundern Sie sich nicht, wenn zunächst Datenmüll auf dem Bildschirm sichtbar wird. Er verschwindet nach dem Laden (<CTRL L>) oder Löschen (<SHIFT CLR/HOME>). Mit den Cursortasten bewegen Sie einen kleinen Cursor auf dem Bildschirm. Text kann wie gewohnt eingegeben werden. Durch zusätzliches Drücken von <SHIFT> werden Buchstaben invers dargestellt. Zur Verfügung stehen folgende Befehle:

Cursortasten
... bewegen den Schriftcursor über den Bildschirm.

<INST/DEL>
... löscht ein Zeichen

<CLR/HOME>
... bringt den Cursor nach links in die oberste Zeile.

<SHIFT CLR/HOME>
... löscht den Bildschirm ohne die Cursorposition zu ändern.

<CTRL L>
... lädt ein File von Diskette. Dabei wird auf die Namenseingabe gewartet. Für Files, die nicht unter HILO gespeichert wurden, muß bei der Eingabe von Buchstaben zusätzlich <SHIFT> gedrückt und ein Sternchen nach dem Namen angehängt werden.

<CTRL S>
... speichert ein Bild auf Diskette. Buchstaben, die ohne <SHIFT> eingegeben werden, erscheinen im Directory Revers.

<CTRL D>
... zeigt das Directory einer eingelegten Diskette.

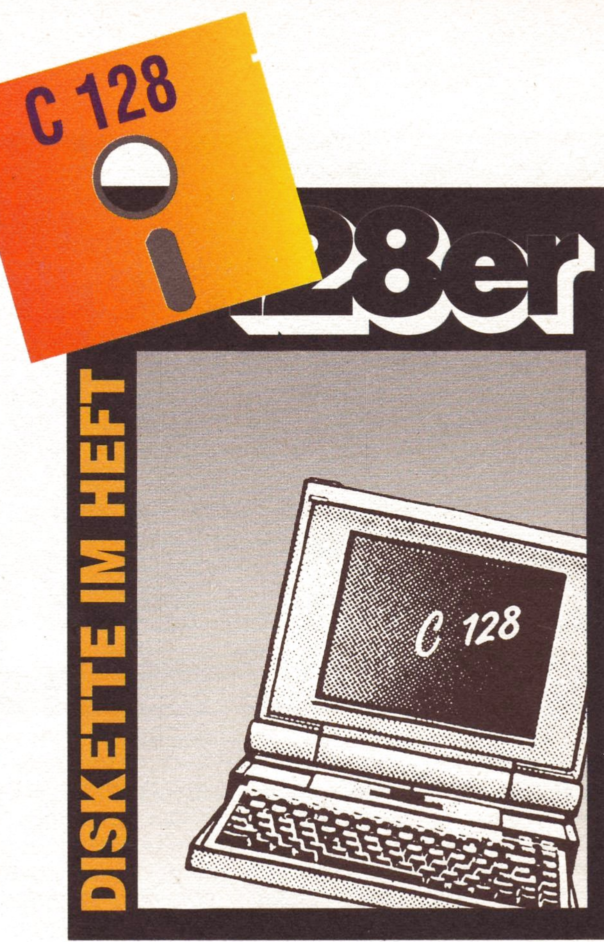
<CTRL I>
... invertiert den Bildschirm

<CTRL O>
... löscht den Bildschirm oberhalb des Cursors. Achtung: der komplette obere Teil wird entfernt.

<CTRL U>
... löscht den Bildschirm unterhalb des Cursors. Achtung: auch hier wird der komplette untere Teil entfernt.

<CTRL E>
... löscht den Bildschirm links vom Cursor. Achtung: Wie bei anderen Löschfunktionen wird die komplette Seite gelöscht.

SONDERHEFT 76 VORSCHAU



Endlich ist's wieder soweit – das nächste Sonderheft gehört exklusiv den C-128-Freunden:

■ – »DiskEtti 128« druckt übersichtliche Diskettenaufkleber – sogar in hochauflösender Grafik!

■ – Die Basic-Erweiterung »Sprite-Tool« macht Unmögliches möglich: Doppelt so viele Sprites auf einem Screen!

■ – Mit »Music-Master V2.0« finden Sie jeden Titel Ihrer Compact-Disc-, LP- und Audiokassettensammlung im Handumdrehen.

■ – Zwei knifflige Knobelspiele beweisen mal wieder, daß man mit dem C 128 eine Menge Spaß haben kann.

■ – Eine umfangreiche Sammlung nützlicher Tips & Tricks unterstützt Sie bei der professionellen Gestaltung eigener Programme.
Unsere CP/M-Freunde erfahren, wo's noch Software gibt!

Aus aktuellen oder technischen Gründen können Themen verschoben werden. Wir bitten dafür um Verständnis.

Das Sonderheft 76 gibt's ab 20.3.1992 bei Ihrem Zeitschriftenhändler.

<**CTRL R**>
... löscht den Bildschirm rechts vom Cursor. Achtung: alles rechts wird entfernt, s. o.
<**CTRL Q**>
... verläßt das Programm. Es kann von BASIC aus mit SYS49152 neu gestartet werden.

Makro

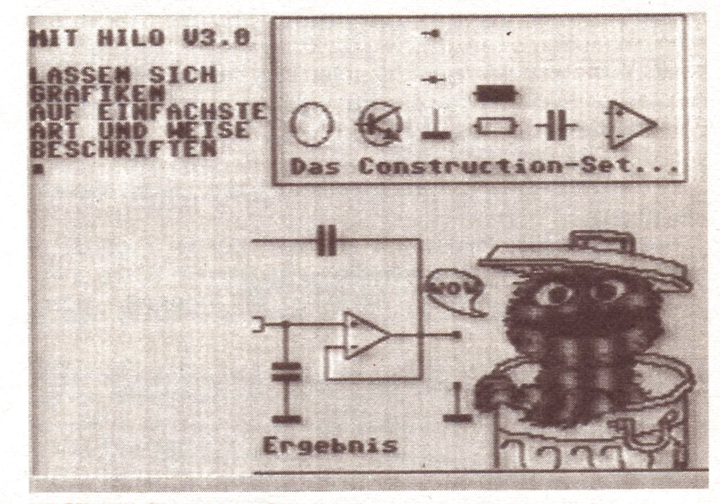
Die Verwendung eines frei am Bildschirm verschiebbaren Makros ermöglichen drei Befehle. Sie werden per Tastendruck aktiviert.
<**CBM O**>
... setzt die linke obere Begrenzung des Makros
<**CBM U**>
... setzt die rechte untere Begrenzung des Makros
<**CBM D**>
... kopiert ein definiertes Makro an die momentane Cursorposition

Über die Speicherstelle 49311 lassen sich einige Presets ändern. Dazu müssen Sie zuerst das Programm verlassen (<CTRL Q>). Danach ändern Sie mit

```
POKE49311,X
```

Dabei kann »X« folgende Werte annehmen:
208 – Grafikzeichen invertiert/normal
210 – Invertierte Großbuchstaben und Kleinschrift
212 – Invertierte Grafikzeichen und Großschrift
214 – Kleinschrift normal und invertiert
216 – Großbuchstaben normal und invertiert

Anschließend läßt sich mit SYS49152 ohne Datenverlust starten. (gr)



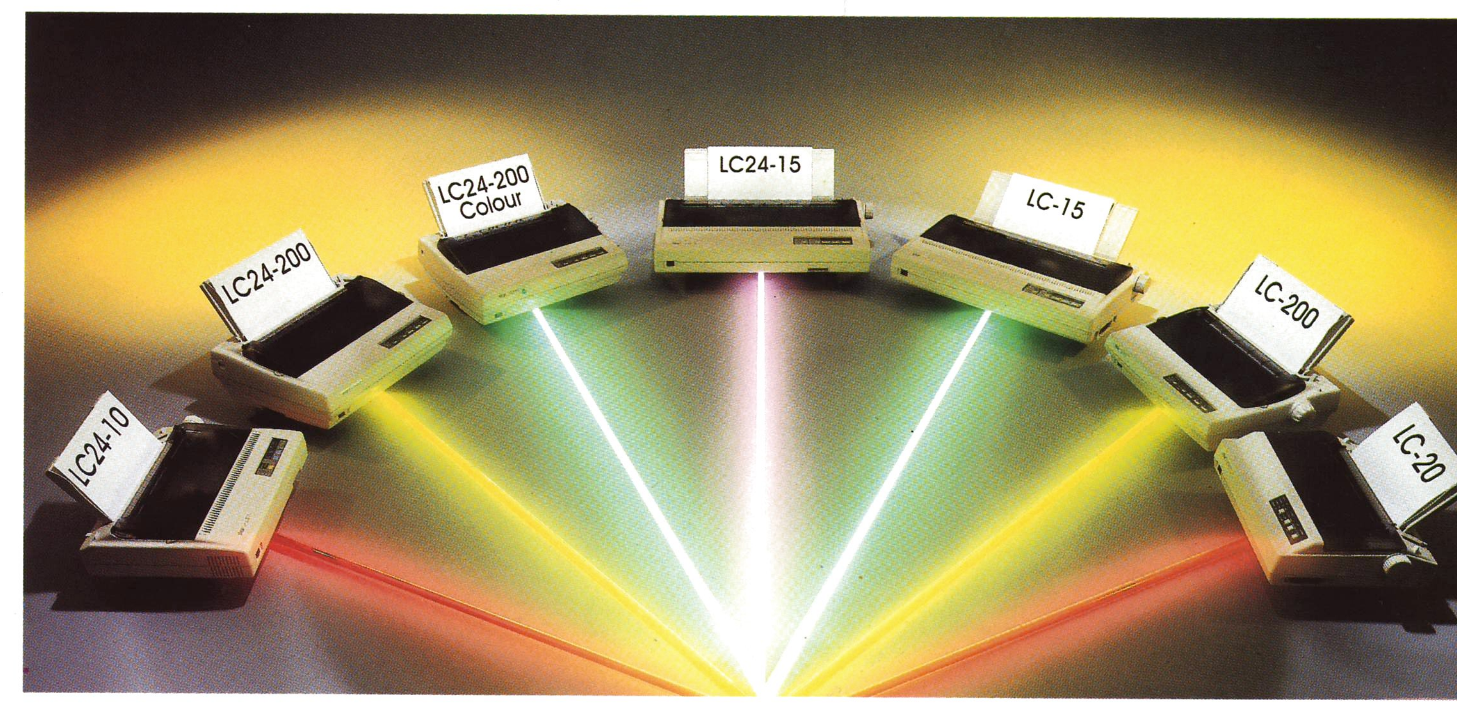
[3] Mit HILO zaubern Sie ohne großen Aufwand nachträglich Texte in Ihre Grafiken

Kurzinfo: HILO V3.0

Programmart: Grafik-Tool
Laden: LOAD"HILO V3.0",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 7
Programmautor: Jörg Brokamp

Attraktiv in Qualität und Preis.

Erfolg in Serie. Star ComputerDrucker



LC24-10

- 24 Nadel-Matrix-Drucker
- LQ- und EDV Ausdrucke (max. 180 cps)
- 4 eingebaute LQ-Schriftarten
- Halbautomatischer Papiereinzug
- Schubtraktor

LC-15

- 9 Nadel-Matrix-Drucker
- NLQ- und EDV Ausdrucke (max. 180 cps)
- 4 eingebaute NLQ-Schriftarten
- Halbautomatischer Papiereinzug
- Papiereinzug auch von unten

LC24-200/LC24-200 Colour

- 24 Nadel-Matrix-Drucker
- LQ-, EDV- und HS-EDV Ausdrucke (max. 222 cps)
- 5 eingebaute LQ-Schriftarten
- Papiereinzug auch von unten
- Kombinierter Zug-/Schubtraktor
- Leise Modus
- LC24-200 Colour: Farbdruck (7 Farben inklusive schwarz)

LC-200

- 9 Nadel-Matrix-Drucker
- NLQ-, EDV- und HS-EDV Ausdrucke (max. 225 cps)
- 4 eingebaute NLQ-Schriftarten
- Papiereinzug auch von unten
- Schaltbarer Zug-/Schubtraktor
- Farbdruck (7Farben inklusive schwarz)

LC24-15

- 24 Nadel-Matrix-Drucker
- LQ- und EDV Ausdrucke (max. 180 cps)
- 4 eingebaute LQ-Schriftarten
- Halbautomatischer Papiereinzug
- Schubtraktor

LC-20

- 9 Nadel-Matrix-Drucker
- NLQ- und EDV Ausdrucke (max. 180 cps)
- 4 eingebaute NLQ-Schriftarten
- Halbautomatischer Papiereinzug



Nähere Informationen über Star ComputerDrucker erhalten Sie beim autorisierten Star Fachhändler oder direkt bei uns.
Star Micronics Deutschland GmbH, Westerbachstr. 59, D-6000 Frankfurt/Main 94, Tel. (069) 78999-0



Power Play -
die ganze Welt der Spiele in einem Heft!
DAS MEISTGEKAUFTE COMPUTER- UND VIDEOSPIELE-MAGAZIN
Markt&Technik
DM 6,50
WER
PLAY
SPIELE-
TESTS
Hol Dir die Nr. 1
der Spiele-Magazine!